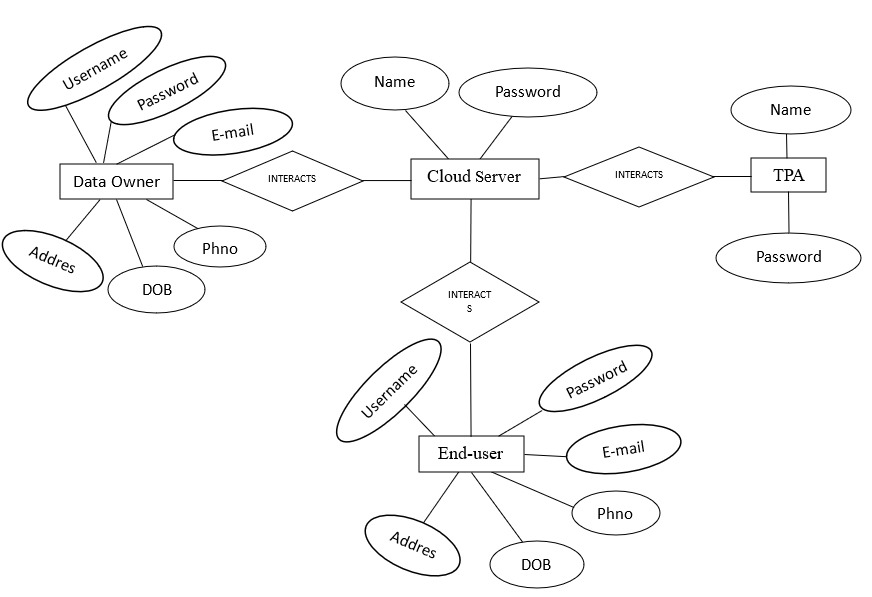
**3. SYSTEM DESIGN**

**3.1 Database Design (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.

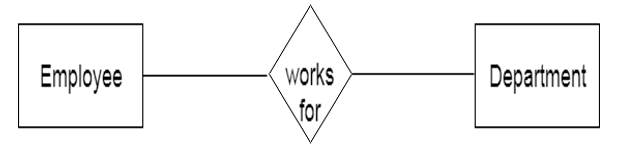
**3.1.1ERmodel**

**Fig 3.1.1.1: ER model Diagram**

1. 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.
2. It establishes a structured framework for the database. Moreover, it provides a straightforward and easily understandable perspective on the data.
3. In Entity-Relationship modeling, the organizational database structure is depicted through a design known as an Entity-Relationship diagram.
4. 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.

### 1**. Entity**

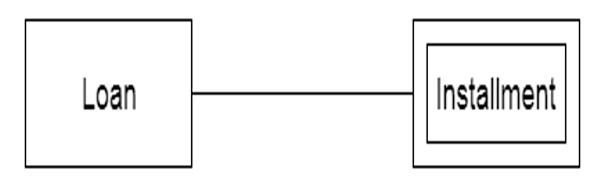
A substance may be anything, class, individual or spot. In the ER frame, a substance can be tended to as square shapes.



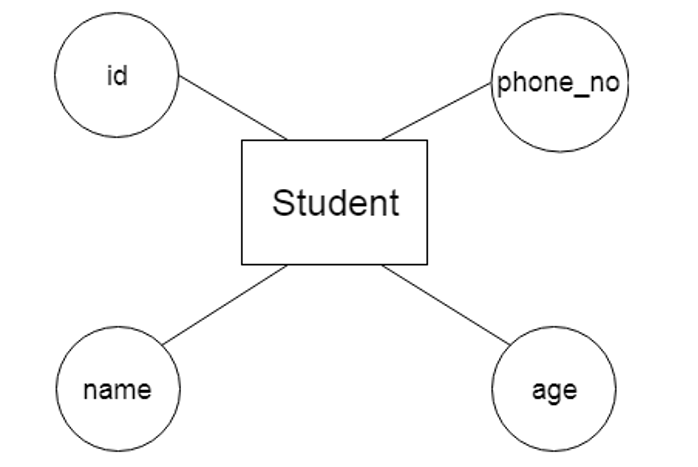
Think about a relationship as a delineation chief, thing, specialist, office, etc can be taken as a substance.

**1.Powerless Entity**

A substance that depends upon another component called a frail substance.

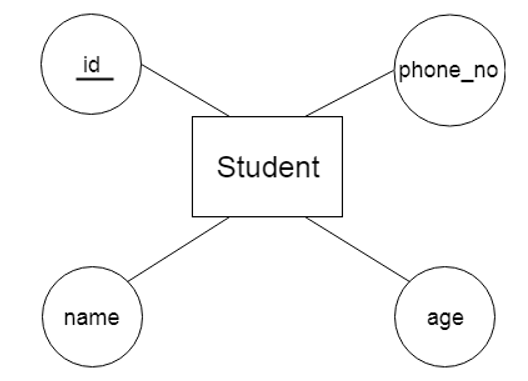


The frail element contains no critical trait of its own. The feeble substance is addressed by a twofold square shape.

1. Characteristic
2. The quality is utilized to depict the property of a section. Obscure is utilized to address
3. a quality.

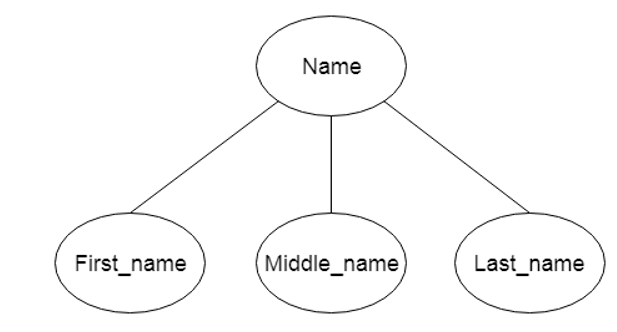
For example, id, age, contact number, name, etc can be attributes of a student.

**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**

****

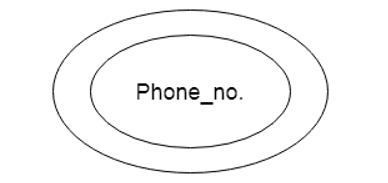
A property that made from various attributes is known as a composite quality.

The composite trademark is tended to by an oval, and those circles are related with a circle**.**

The composite trademark is tended to by an oval, and those circles are related with a circle.

**c. Multivalued Attribute**

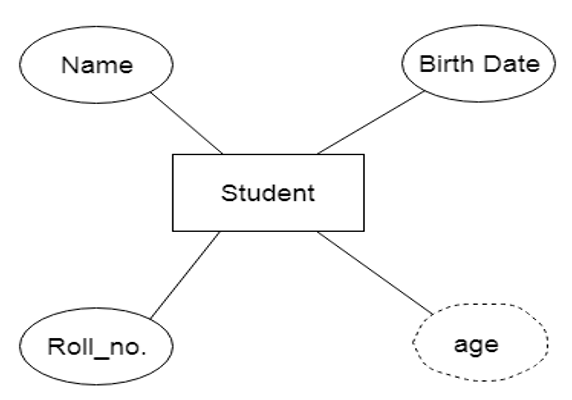
A quality can have more than one worth. These qualities are known as a multivalued property.



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**

Teacher

Student

Teaches

1

M

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:**

1. **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.

Sign up

Tweet Server

Login

1

M

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.

Tweet Server

Remote User

has

1

M

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.

User

Service Provider

Request

1

M

For example, Student enrols 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.

Users

Logins

has

1

M

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

**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.

Table name: Data owner

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Type** | **Collation** | **Attributes** | **Null** | **Default** |
| User\_name | Int(11) | Latin\_swedish\_ci |  | No | None |
| Password | Varchar(30) | Latin\_swedish\_ci |  | No | None |
| E\_mail | Varchar(30) | Latin\_swedish\_ci |  | No |  |
| Ph\_No | Varchar(30) | Latin\_swedish\_ci |  | No |  |
| DOB | Varchar(30) | Latin\_swedish\_ci |  | No |  |
| Address | Varchar(30) | Latin\_swedish\_ci |  | No |  |

**Table 3.2.2 Data owner**

Table name: End User

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Type** | **Collation** | **Attributes** | **Null** | **Default** |
| User\_name | Int(11) | Latin\_swedish\_ci |  | No | None |
| Password | Varchar(30) | Latin\_swedish\_ci |  | No | None |
| E\_mail | Varchar(30) | Latin\_swedish\_ci |  | No |  |
| Ph\_No | Varchar(30) | Latin\_swedish\_ci |  | No |  |
| DOB | Varchar(30) | Latin\_swedish\_ci |  | No |  |
| Address | Varchar(30) | Latin\_swedish\_ci |  | No |  |

**Table 3.2.3 Enduser**

Table name: Cloud Server

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Type** | **Collation** | **Attributes** | **Null** | **Default** |
| User\_name | Int(11) | Latin\_swedish\_ci |  | No | None |
| Password | Varchar(30) | Latin\_swedish\_ci |  | No | None |

**Table 3.2.4 Cloud Server**

Table name: TPA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Type** | **Collation** | **Attributes** | **Null** | **Default** |
| User\_name | Int(11) | Latin\_swedish\_ci |  | No | None |
| Password | Varchar(30) | Latin\_swedish\_ci |  | No | None |

**Table 3.2.4 TPA**

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

**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** | **Username** | **Password** |
| 015 | John | \*\*\*\*\*\*\*\* |
| 016 | Princess | \*\*\*\*\*\*\*\* |
| 027 | Tom | \*\*\*\*\*\*\*\* |
| 028 | Claire | \*\*\*\*\*\*\*\* |
| 029 | Robert | \*\*\*\*\*\*\*\* |

**Table No 3.2.5. 1NF**

**Second typical structure (2NF)**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 No 3.3.2.6 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).

**Third ordinary structure (3NF)**

USERDETAILS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **NAME** | **EMAIL** | **STATE** | **CITY** | **COUNTRY** |
| 11 | Vijay | [vijay@gmail.com](mailto:vijay@gmail.com) | AP | RZP | INDIA |
| 12 | Vinod | [vinod@gmail.com](mailto:vinod@gmail.com) | AP | RZP | INDIA |
| 13 | Ramu | Ramu@gmail.com | AP | RZP | INDIA |
| 14 | Vishnu | vishnu@gmail.com | AP | RZP | INDIA |

**Table no 3.2.7 User Details**

USER DETAILS

|  |  |  |
| --- | --- | --- |
| **USER ID** | **PASSSWORD** | **LOGIN** |
| server | \*\*\*\*\*\*\*\* | Sign\_up |
| Vijay | \*\*\*\*\*\*\*\* | Sign\_up |

**Table No 3.2.8. User details**

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:

**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. These normal forms help structure tables efficiently and reduce redundancy and inconsistency in data.

**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. It requires that for every non-trivial functional dependency (X → Y), X must be a superkey.

**Fourth Normal Form (4NF):** To achieve 4NF, a table must be in BCNF and should not have multi-valued dependencies.

**Essential Tuple Normal Form (ETNF):** ETNF is a condition where each attribute in a tuple is essential to the understanding of the tuple itself.

|  |  |
| --- | --- |
| **Normal Form** | **Description** |
| [1NF](https://www.javatpoint.com/dbms-first-normal-form) | An alliance is in 1NF enduring it contains an atomic worth. |
| [2NF](https://www.javatpoint.com/dbms-second-normal-form) | 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](https://www.javatpoint.com/dbms-third-normal-form) | 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](https://www.javatpoint.com/dbms-forth-normal-form) | An association will be in 4NF expecting it is in Boyce Codd's commonplace construction and has no multi-regarded dependence. |
| [5NF](https://www.javatpoint.com/dbms-fifth-normal-form) | 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.

**3.3 UML DIAGRAMS**

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntax, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective.

UML is specifically constructed through two different domains they are:

* UML Analysis modeling, this focuses on the user model and structural model views of the system.
* UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

**System Design Aspects**

Once the analysis stage is completed, the next stage is to determine in broad outline form how the problem might be solved. During system design, we are beginning to move from the logical to physical level.

System design involves architectural and detailed design of the system. Architectural design involves identifying software components, decomposing them into processing modules and conceptual data structures, and specifying the interconnections among components.

Detailed design is concerned with how to package processing modules and how to implement the processing algorithms, data structures and interconnections of standard algorithms, invention of new algorithms, and design of data representations and packaging of software products. Two kinds of approaches are available:

* Top-down approach
* Bottom-up approach

**Design of Code**

Since information systems projects are designed with space, time and cost saving in mind, coding methods in which conditions, words, ideas or control errors and speed the entire process. The purpose of the code is to facilitate the identification and retrieval of the information. A code is an ordered collection of symbols designed to provide unique identification of an entity or an attribute.

**Design of Input**

Design of input involves the following decisions

* Input data
* Input medium
* The way data should be arranged or coded
* Validation needed to detect every step to follow when error occurs

The input controls provide ways to ensure that only authorized users access the system guarantee the valid transactions, validate the data for accuracy and determine whether any necessary data has been omitted. The primary input medium chosen is display. Screens have been developed for input of data using HTML. The validations for all important inputs are taken care of through various events using JSP control.

**Design of Output**

Design of output involves the following decisions

* Information to present
* Output medium
* Output layout

Output of this system is given in easily understandable, user-friendly manner, Layout of the output is decided through the discussions with the different users.

**Design of Control**

The system should offer the means of detecting and handling errors.

Input controls provides ways per

* Valid transactions are only acceptable
* Validates the accuracy of data
* Ensures that all mandatory data have been captured

All entities to the system will be validated. And updating of tables is allowed for only valid entries. Means have been provided to correct, if any by change incorrect entries have been entered into the system they can be edited.

**Why We Use UML in projects?**

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modeling Language (UML) was designed to respond to these needs. Simply, Systems design refers to the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements which can be done easily through UML diagrams.

In the project four basic UML diagrams have been explained among the following list:

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

**Class Diagram**

A Class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

This is one of the most important of the diagrams in development. The diagram breaks the class into three layers. One has the name, the second describes its attributes and the third its methods. A padlock to left of the name represents the private attributes. The relationships are drawn between the classes. Developers use the Class Diagram to develop the classes. Analyses use it to show the details of the system.

Architects look at class diagrams to see if any class has too many functions and see if they are required to be split.



Fig : 3.3.1. Class diagram

**Use Case Diagram**

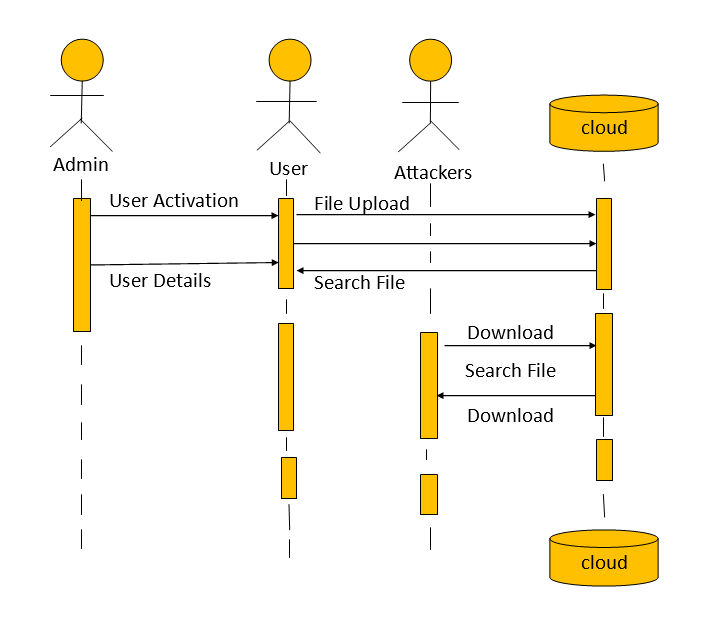
A Use Case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. 

Fig: 3.3.2. Use Case Diagram

Use cases focus on the behavior of the system from the external point of view. The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system

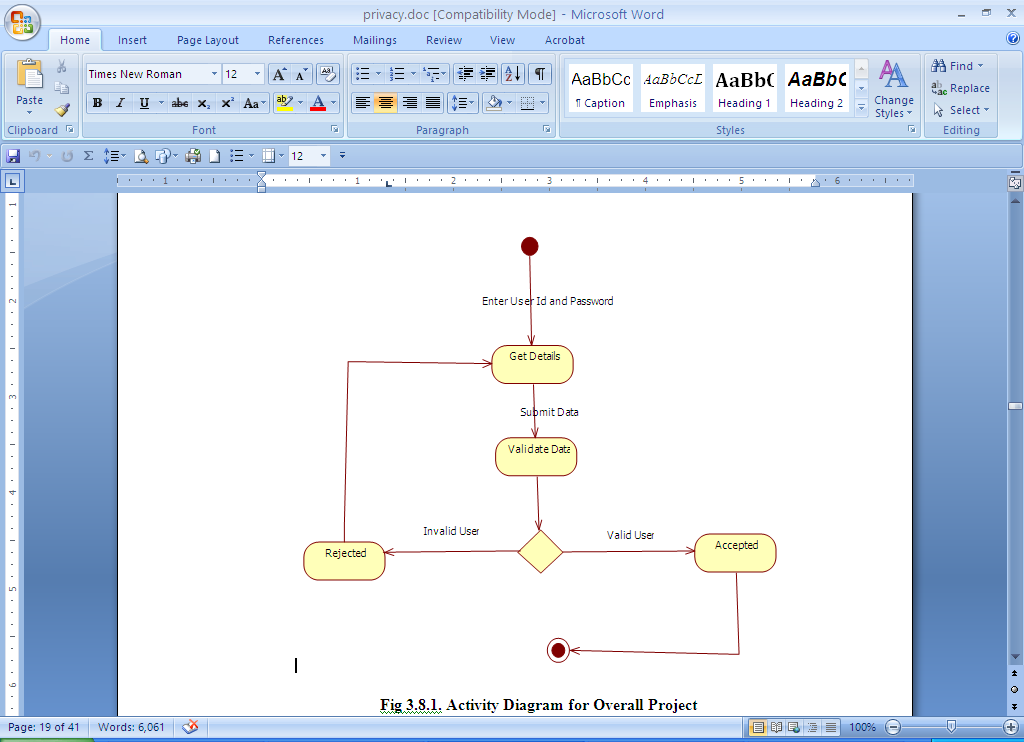
**Sequence Diagram**

A Sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams

****

**Fig : 3.3.3 sequence diagram**

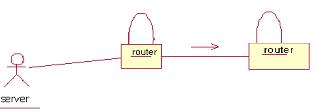
**Activity Diagram**

Activity diagrams are a loosely defined diagram technique for showing workflows of stepwise activities and actions, with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**Fig:3.3.4. Activity diagram**

**Collaboration Diagram**

A Communication diagram models the interactions between objects or parts in terms of sequenced messages. Communication diagrams represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behavior of a system.



**Fig:3.3.5 Collaboration diagram**

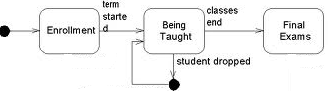
**Deployment Diagram**

****A Deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected e.g. JDBC, REST

**Fig :3.3.5. Deployment diagram**

**State Chart Diagram**

A State diagram is a type of diagram used in computer science and related fields to describe the behavior 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.



**Fig.3.7: State Chart Diagram**

**Component Diagram**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.



**Fig.3.8: Component Diagram**

**DATA FLOW DIAGRAM**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

#### Define source and destination data.

#### Shows path of the data flow.

#### To represent a process that transforms or modifies the Data

To represent an attribute

|  |  |
| --- | --- |
|  |  |

Data Store



**Fig.3.9: Data Flow Diagram**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns

and components.

1. Integrate best practices.