**4. SYSTEM DESIGN**

**4.1. 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 syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* **User Model View:**

1. This view represents the system from the users perspective.
2. The analysis representation describes a usage scenario from the end-users perspective.

* **Structural Model view:**

1. In this model the data and functionality are arrived from inside the system.
2. This model view models the static structures.

* **Behavioral Model View:** It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.
* **Implementation Model View:** In this the structural and behavioral as parts of the system are represented as they are to be built.
* **Environmental Model View:** In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.
* **Types Of UML Diagrams:**
* Use Case Diagram
* Class Diagram
* Sequence Diagram
* Collaboration Diagram
* Component Diagram
* Deployment Diagram
* Activity Diagram

**4.1.1. USE CASE DIAGRAM**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**Figure 4.1.1:** Use Case Diagram

**4.1.2. CLASS DIAGRAM**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake



**Figure 4.1.2:** Class Diagram

**4.1.3. SEQUENCE DIAGRAM**

A **Sequence Diagram** 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. 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 typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.



**Figure 4.1.3:** Sequence Diagram

**4.1.4. COLLABORATION DIAGRAM**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration 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.



**Figure 4.1.4:** Collaboration Diagram

**4.1.5 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.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

  
**Figure 4.1.5:** Component Diagram

**4.1.6. 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, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



**Figure 4.1.6:** Deployment Diagram

**4.1.7 ACTIVITY DIAGRAM**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

Register

Is Existed User?

Yes No

Login

Upload the file in application layer

View total chunks and their sizes

View the file info from logical layer

Download file from application layer

View metadata comparison chart

Logout

**Figure 4.1.7:** Activity Diagram

**4.2. DATA FLOW DIAGRAM**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

New user has to register

Register

User

Reg. process completed

User can download file from cloud & saved in drive

Enter user, pass to login into the screen

UserScreen

Select any file,Upload it,Uploaded file splitted into chunks with size

Database

**Figure 4.2:** Data Flow Diagram