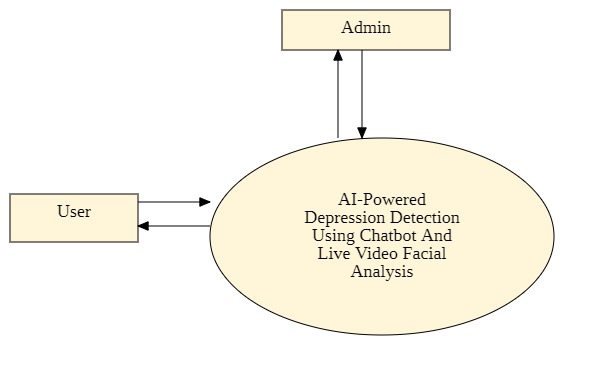
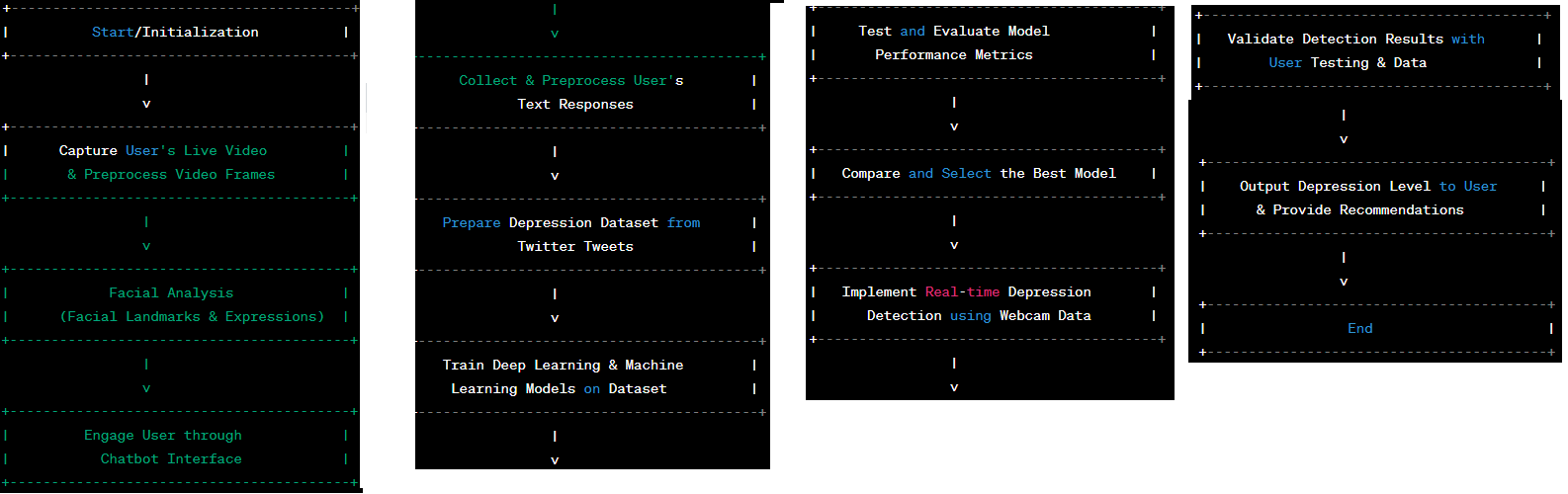
**SYSTEM DESIGN**

**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.



****

**Flow chart Diagram**

* **Start/Initialization:**
  + This step marks the beginning of the project.
  + It involves initializing all the necessary components and systems required for depression detection.
* **Capture User's Live Video & Preprocess Video Frames:**
  + In this step, the system captures live video from the user's webcam.
  + The captured video frames are preprocessed to enhance the quality and reduce noise.
* **Facial Analysis (Facial Landmarks & Expressions):**
  + The preprocessed video frames undergo facial analysis.
  + Facial landmarks are detected, which are specific points on the face (e.g., eyes, mouth) used for analysis.
  + Facial expressions are also analyzed to identify signs of depression (e.g., sadness, lack of expression).
* **Engage User through Chatbot Interface:**
  + The system engages the user through a chatbot interface.
  + This involves initiating a natural language conversation with the user to assess their emotional state and gather information.
* **Collect & Preprocess User's Text Responses:**
  + User's text responses in the chat conversation are collected.
  + These responses are preprocessed to remove noise and irrelevant information.
* **Prepare Depression Dataset from Twitter Tweets:**
  + Depression-related tweets are collected from Twitter.
  + The tweet data is preprocessed to extract relevant features (e.g., sentiment, keywords) for training the model.
* **Train Deep Learning & Machine Learning Models on Dataset:**
  + The prepared depression dataset is used to train various machine learning models.
  + These models may include Support Vector Machines (SVM), Decision Trees, Neural Networks, etc.
  + The models learn to recognize patterns indicative of depression in the text data.
* **Test and Evaluate Model Performance Metrics:**
  + The trained models are tested using a separate dataset.
  + Performance metrics (e.g., accuracy, precision, recall) are computed to assess how well the models can detect depression.
* **Compare and Select the Best Model:**
  + The results of each model's performance are compared.
  + The model with the highest accuracy and the best overall performance is selected for further use.
* **Implement Real-time Depression Detection using Webcam Data:**
  + The selected model is integrated into the live system.
  + Real-time depression detection is performed on user data, combining facial analysis and chatbot responses.
* **Validate Detection Results with User Testing & Data:**
  + User testing is conducted with participants.
  + Self-reported depression levels are collected from users.
  + The system's output is compared with self-reports to validate its accuracy.
* **Output Depression Level to User & Provide Recommendations:**
  + The system provides the detected depression level to the user.
  + Depending on the level detected, appropriate recommendations or support resources may be offered.
* **End:**
  + This marks the conclusion of the AI-Powered Depression Detection process.

**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.

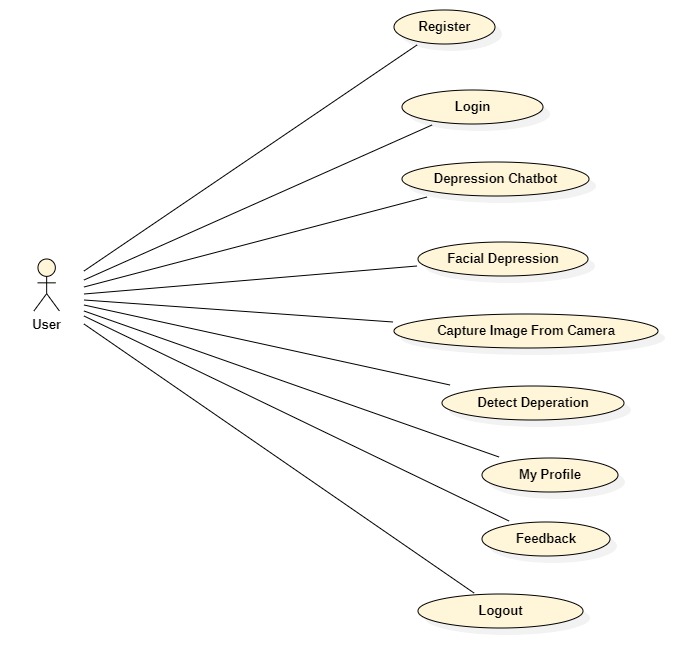
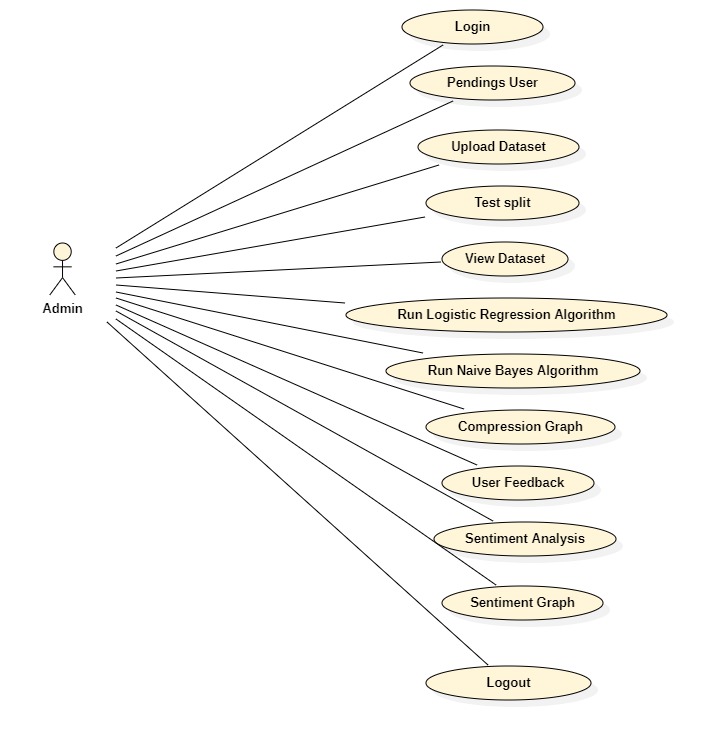
**GOALS:**

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.
7. Integrate best practices.

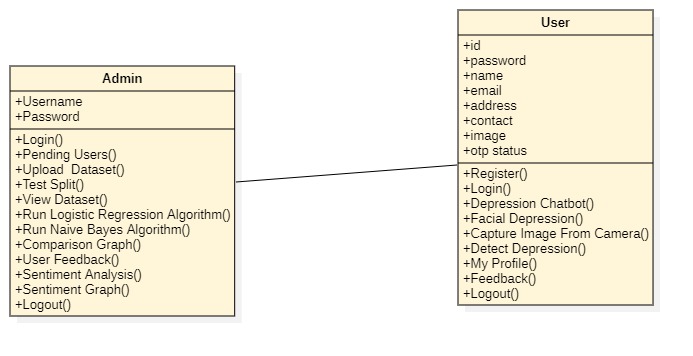
**USE CASE DIAGRAM:**

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.



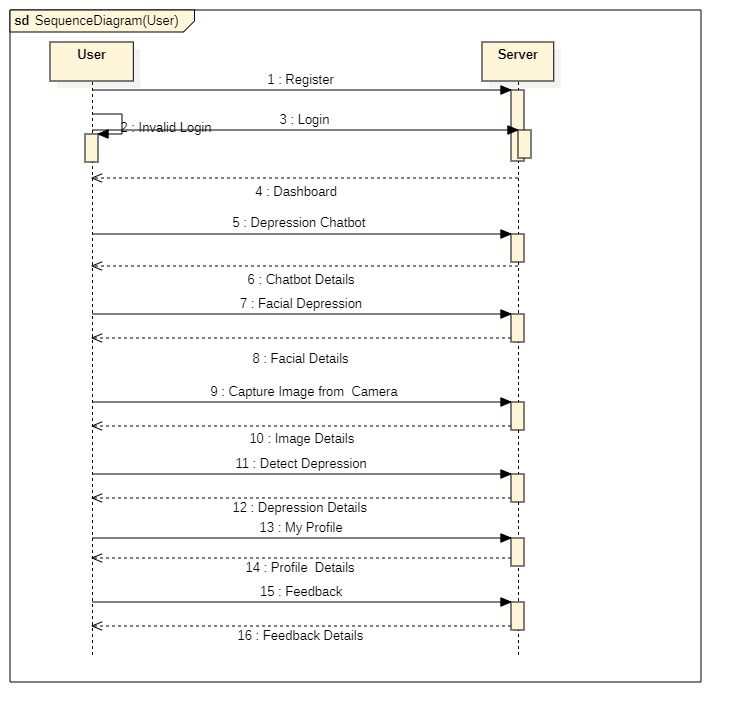
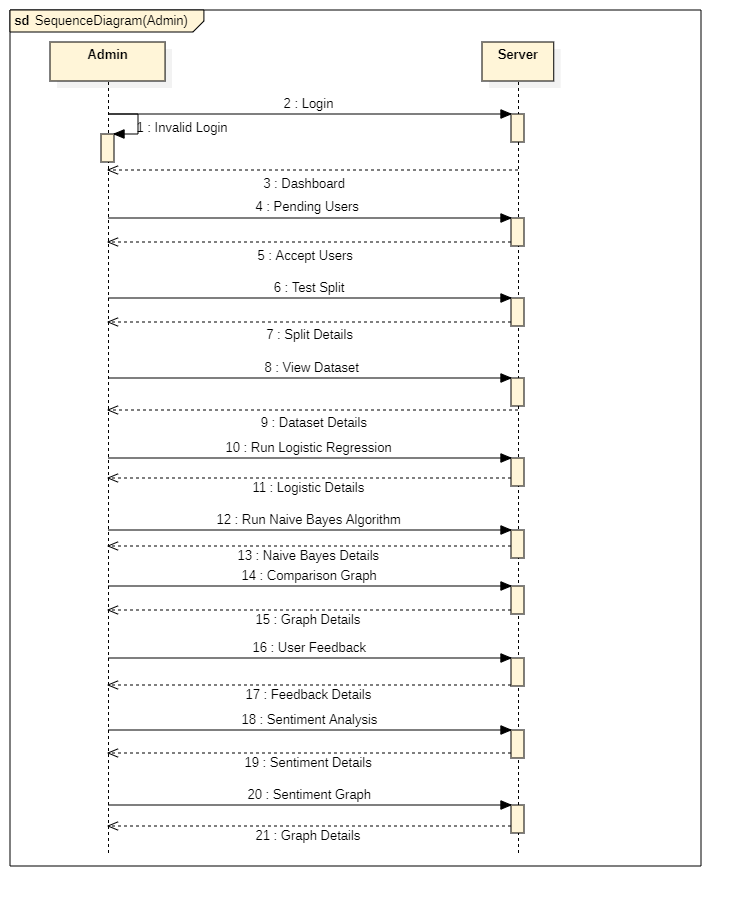
**CLASS DIAGRAM:**

In software engineering, 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, operations (or methods), and the relationships among the classes. It explains which class contains information.



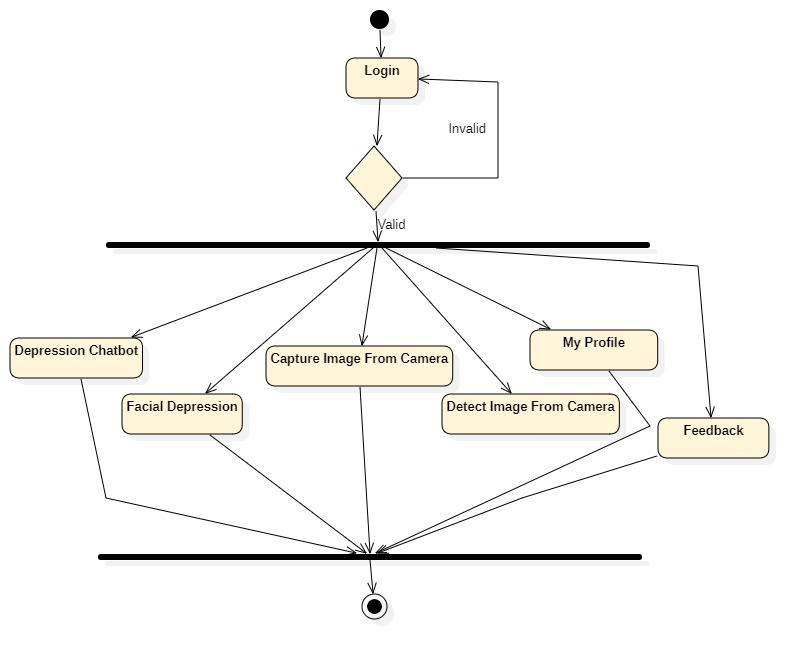
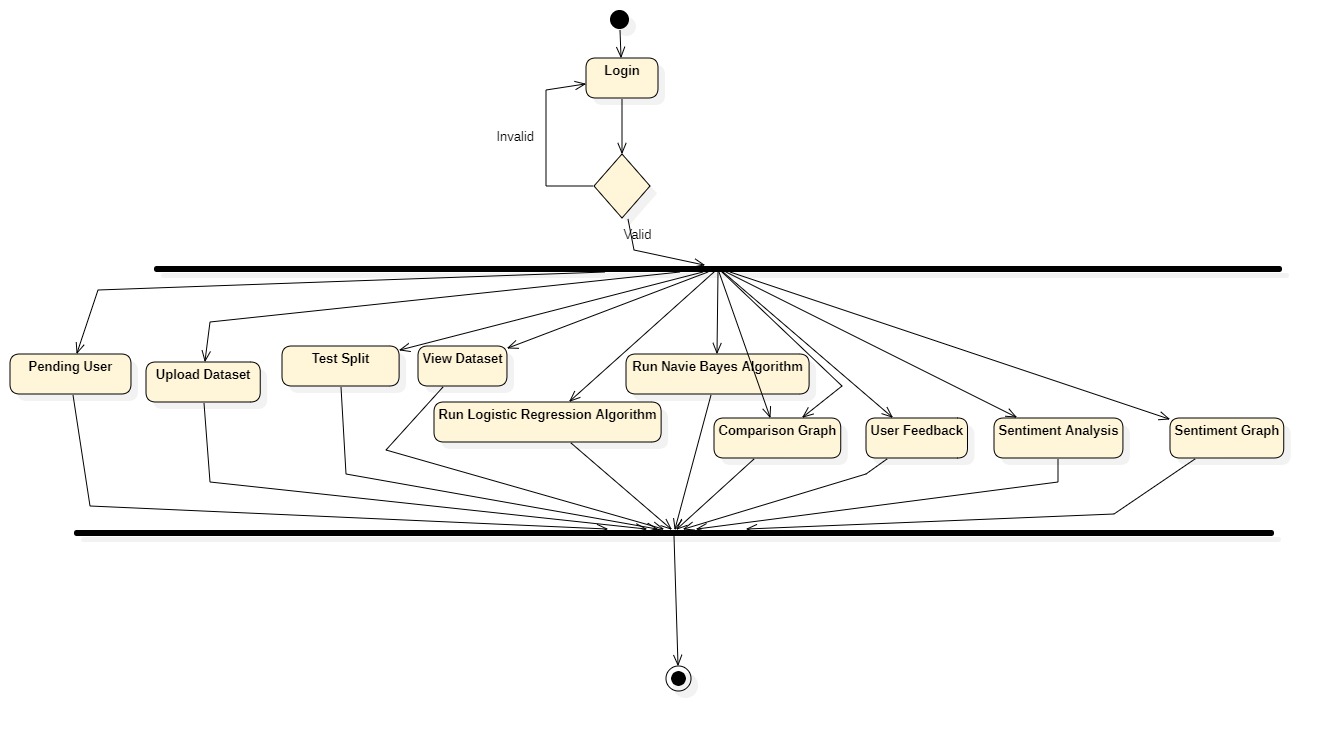
**SEQUENCE DIAGRAM:**

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 diagrams, event scenarios, and timing diagrams.



**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of 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.



**DEPLOYMENT DIAGRAM:**

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

The deployment diagram maps the software architecture created in design to the physical system architecture that executes it. In distributed systems, it models the distribution of the software across the physical nodes.

The software systems are manifested using various artifacts, and then they are mapped to the execution environment that is going to execute the software such as nodes. Many nodes are involved in the deployment diagram; hence, the relation between them is represented using communication paths.

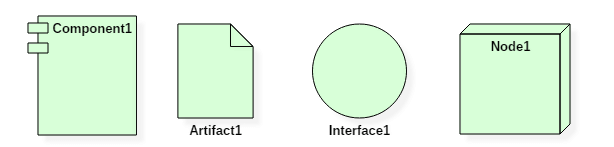
**There are two forms of a deployment diagram.**

* Descriptor form
* It contains nodes, the relationship between nodes and artifacts.
* Instance form
* It contains node instance, the relationship between node instances and artifact instance.
* An underlined name represents node instances.

**Purpose of a deployment diagram**

Deployment diagrams are used with the sole purpose of describing how software is deployed into the hardware system. It visualizes how software interacts with the hardware to execute the complete functionality. It is used to describe software to hardware interaction and vice versa.

**Deployment Diagram Symbol and notations**

Deployment Diagram Notations

