Design diagrams : School Management System

1. Use case diagram:

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Website. Use case diagrams are employed in UML (Unified Modeling Language), standard notation for the modeling of real-world objects and systems.

System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service-oriented task. For example, use cases in a product sales environment would include item ordering, catalog updating, payment processing, and customer relations.

A use case diagram contains four components:

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases. This includes three types as below.
    - Uses or includes
    - Extends
    - Communication

For the process of drawing the use-cases, a clean observation of the system is done and then the processes involved are listed out. Then for each of the process, which all actors are involved is evaluated and then they are linked correspondingly. The actors are represented with a human symbol and its name assigned next to it. Use-cases are represented in an oval shaped box. The relations between the use cases is also analyzed and indicated.

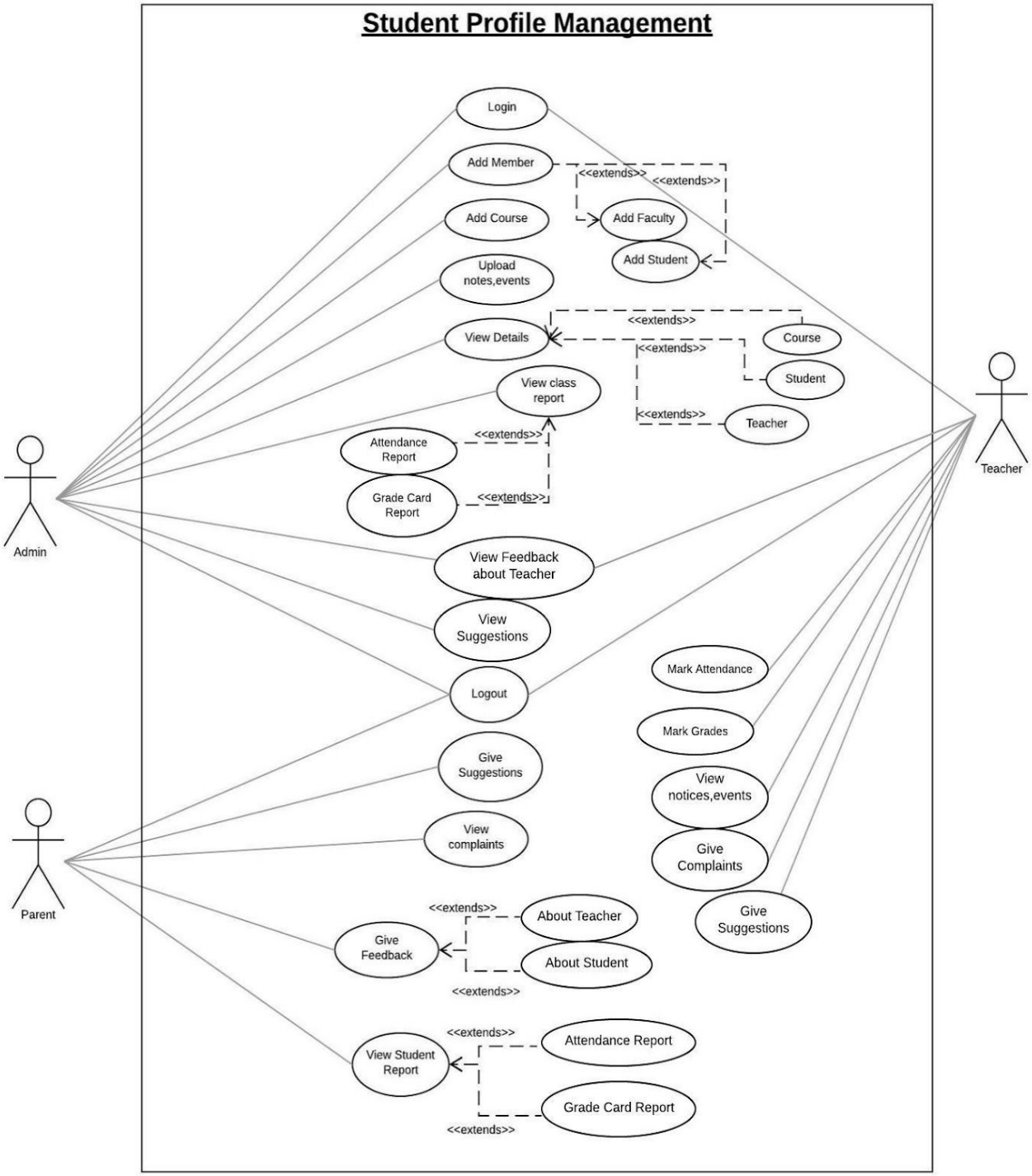


Fig1. Use-Case Diagram

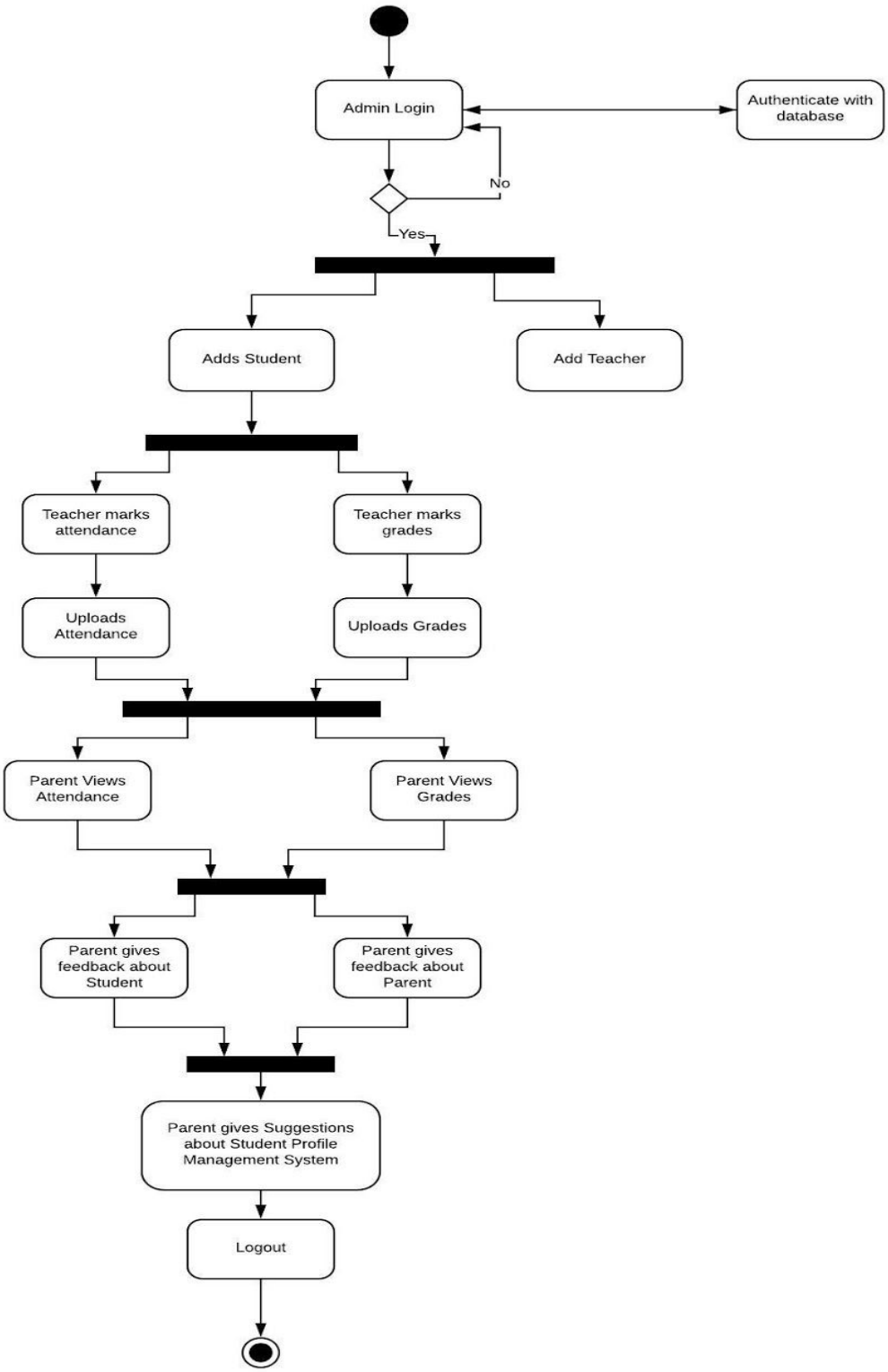
2. Activity diagram:

An activity diagram is used to model a large activity's sequential workflow by focusing on action sequences and respective action initiating conditions. The state of activity relates to the performance of each workflow step. An activity diagram is represented by shapes that are connected by arrows.

- Arrows run from activity start to completion and represent the sequential order of performed activities.
- Black circles represent an initial workflow state.
- A circled black circle indicates an end state.
- Rounded rectangles represent performed actions, which are described by text inside each rectangle.
- A diamond shape is used to represent a decision, which is a key activity diagram concept.

Upon activity completion, a transition (or set of sequential activities) must be selected from a set of alternative transitions for all use cases. Synchronization bars called fork and join indicating the start or completion of concurrent activities are used to represent parallel sub-flows. The activity diagram for the management of attendance and grades throughout the system is given. Note that all the functionalities are not included in this diagram and only the selected states of the whole system which contribute to the grades and attendance management are given in the following diagram.

Fig2. Activity Diagram for grades and attendance



3. Sequence diagram:

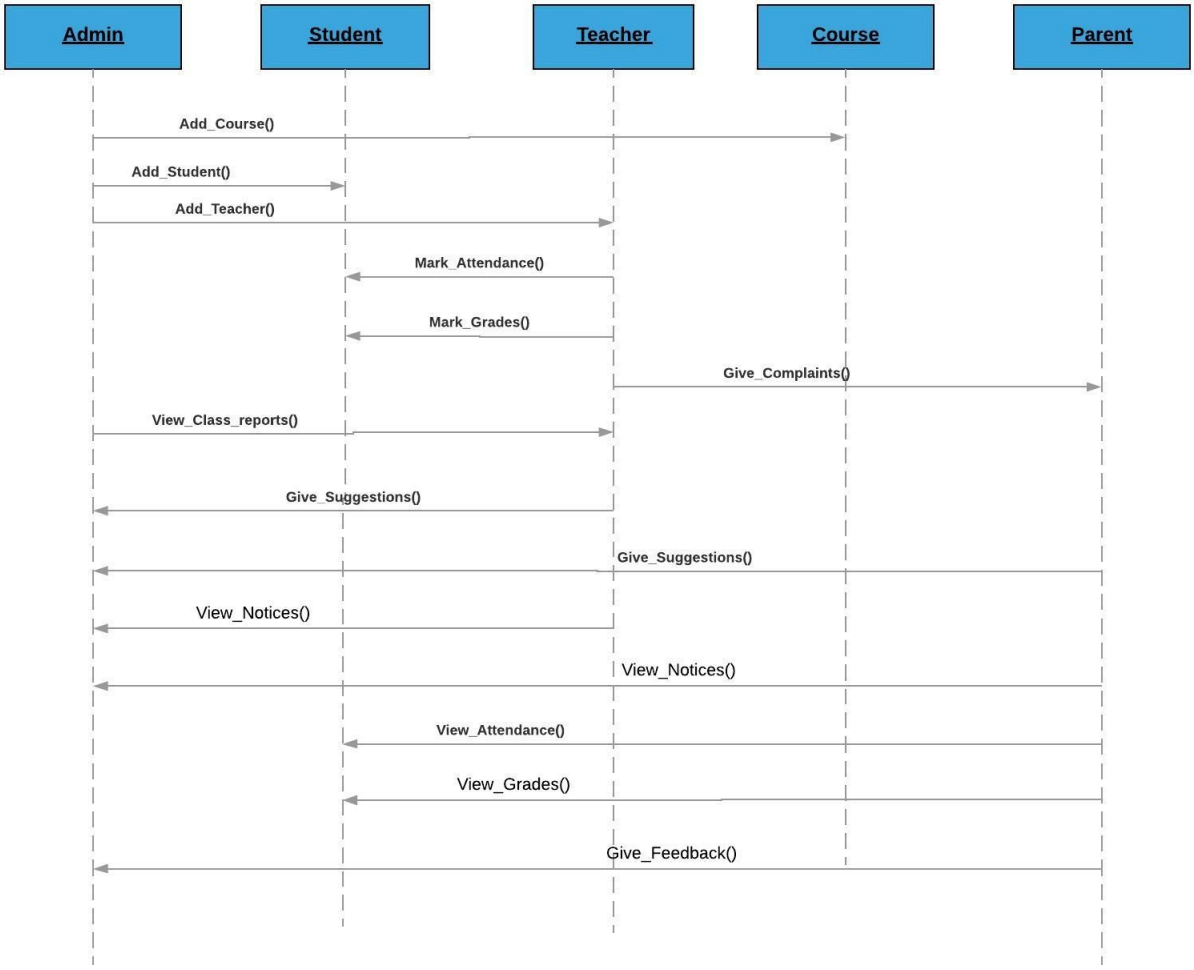
A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

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.

In the Student Profile Management System, some of the functionalities which occur in a particular series in an obvious way are selected and the sequence diagram is limited to only those functions.
The objects listed out for this scenario are as follows.

- Admin
- Student
- Teacher
- Course and
- Parent

Fig3. Sequence Diagram for desired functionalities



4. Class diagram:

The Class Diagram is the main building block of object-oriented modeling. It is used 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 elements, interactions in the application, and the classes to be programmed.
In the diagram, classes are represented with boxes that contain three compartments:

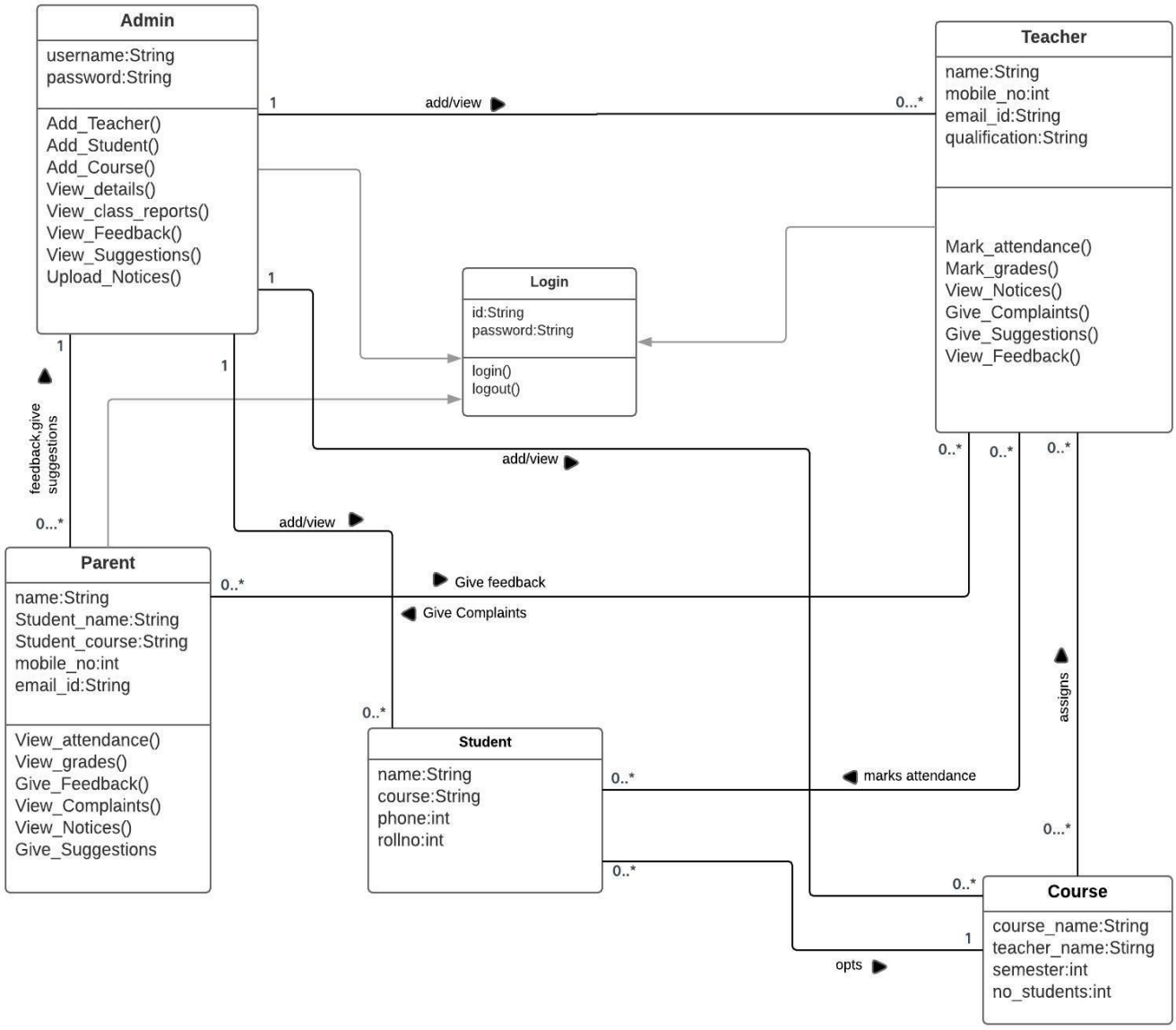
- The top compartment contains the name of the class.
- The middle compartment contains the attributes of the class.
- The bottom compartment contains the operations the class can execute.

In our Scenario classes are identified by Use-Case driven approach. Initially, all the processes which can occur are listed out and actors involving those processes are identified and then finalized into relevant number of classes which can explain all the components in the scenario.
The finalized classes are as below:

- Admin class
- Teacher class
- Parent class
- Student class
- Course class
- Login class

Each of the class has its name written on top, also all the attributes associated with that class in the lower box along with their types and finally the methods which are capable of being performed by the class. All the classes are checked for their association with other classes and it's being mentioned aligned next to the line joining the classes. Multiplicity is indicated at the connecting points of the lines.

Fig4. UML- Class Diagram



5. Data flow diagram:

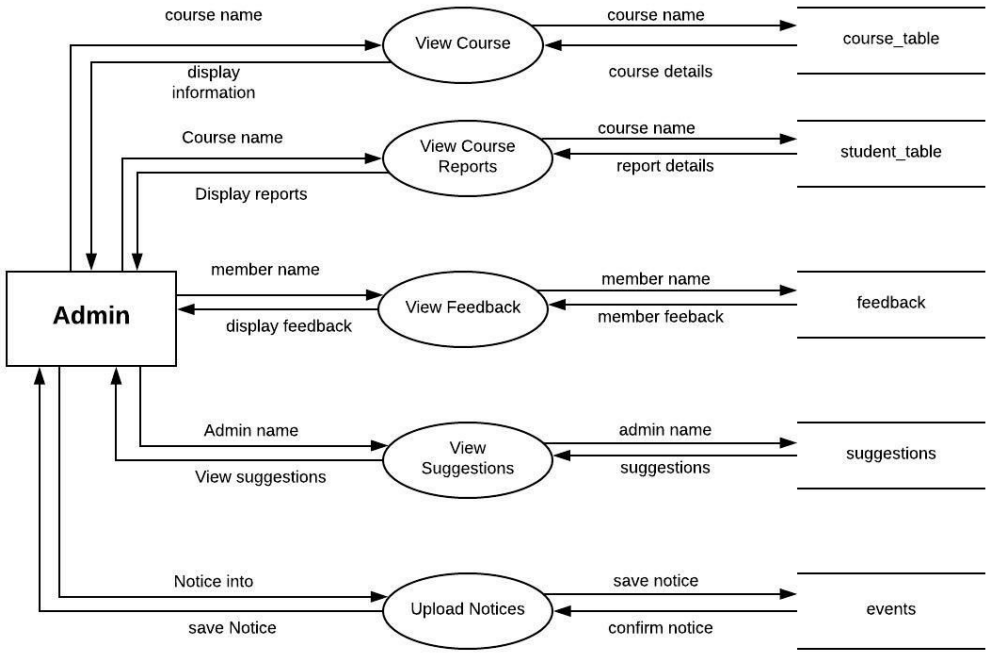
A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data, flows as a unified model. Data Flow diagram contains only 4 components. They are described below.

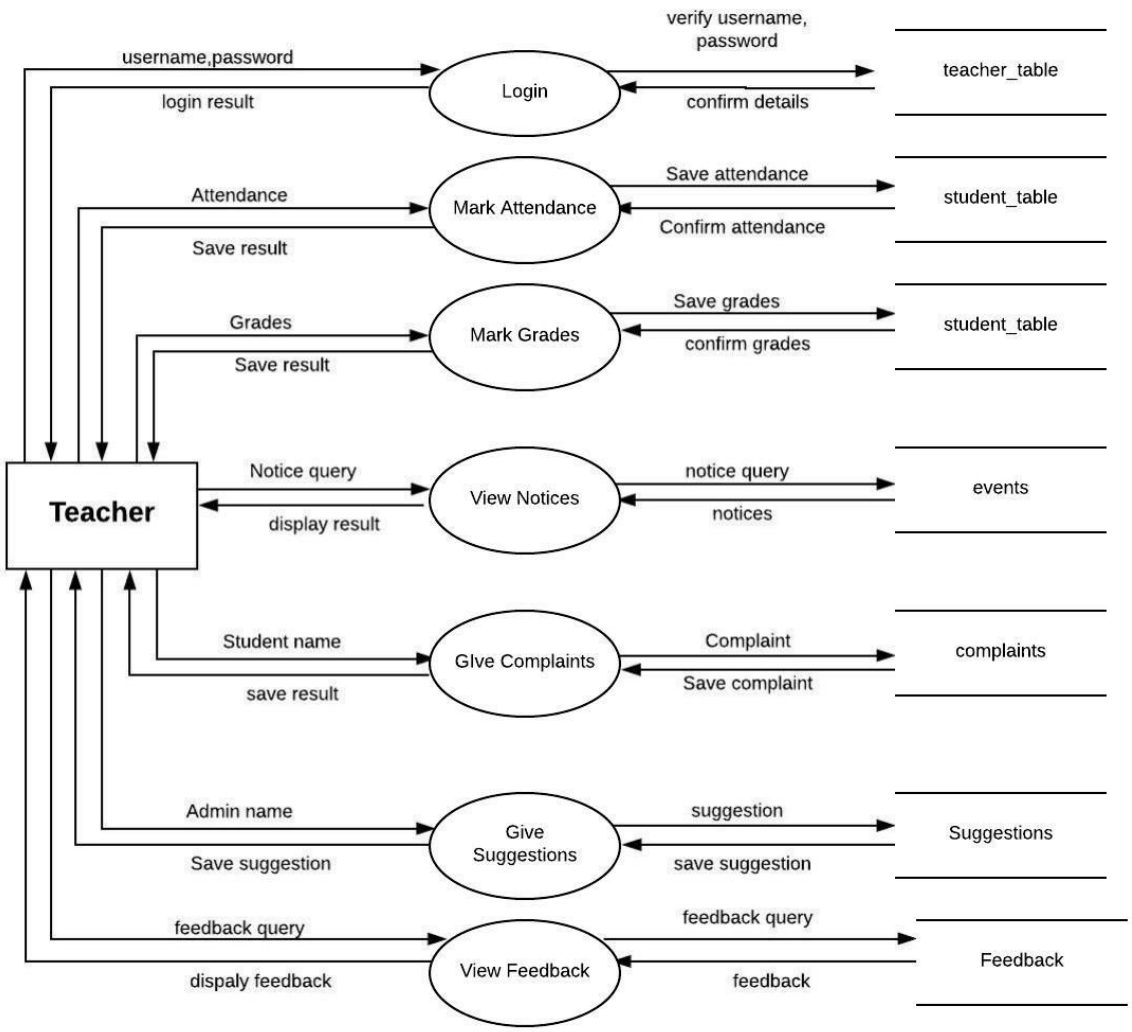
- An External Entity is a source or destination of a data element.
- The Data Store is a location where data is stored.
- Data Flow shows the direction of the data element movement.
- The Process is any function being performed.

For every entity of the system the data flow associated with it is represented in a separate diagram, as there's no cross-communication between the entities and every data element is directed to the data store in this case. The data flow is given for each and every function of entities as decided before and corresponding data store is mentioned in the diagram.

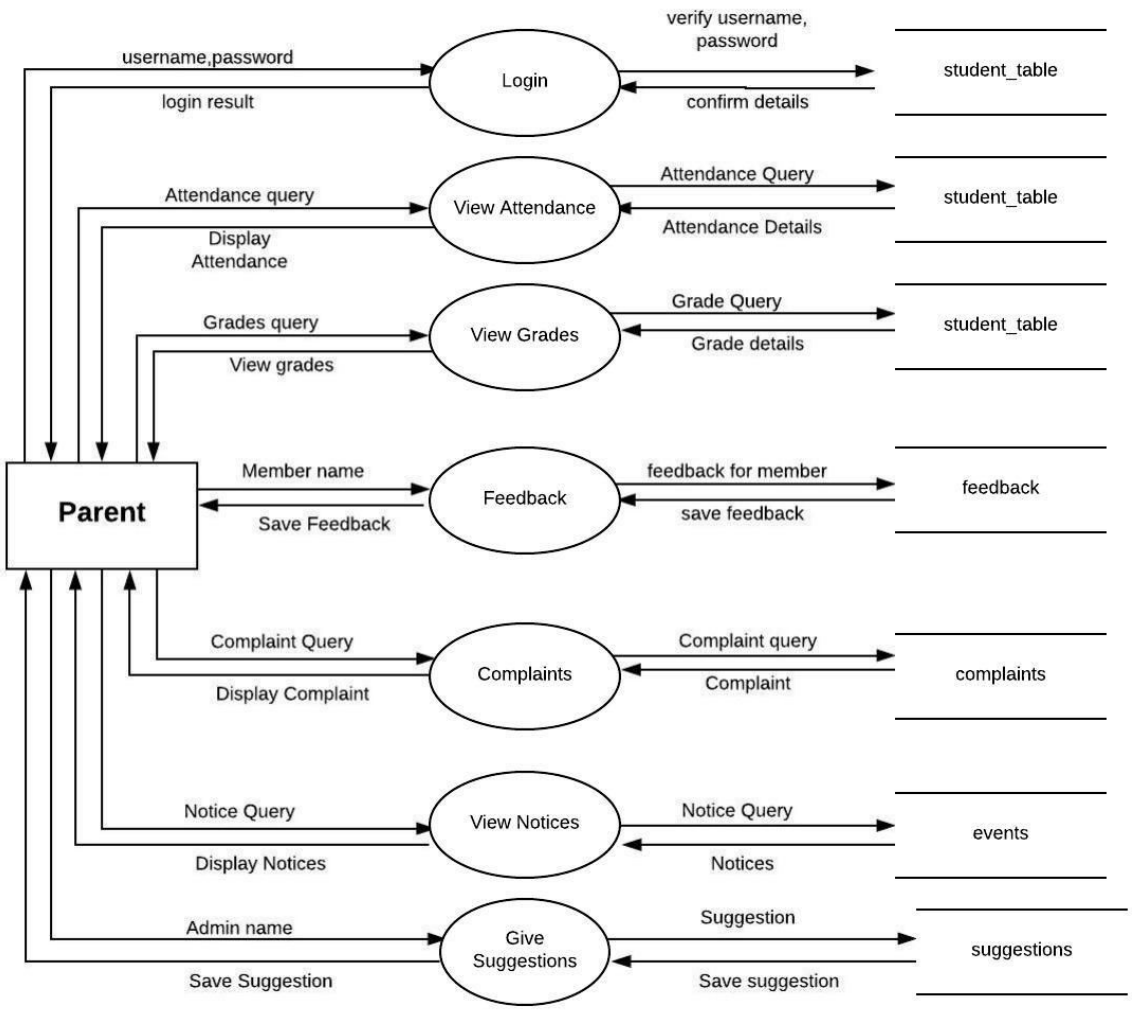
5.1 Data flow diagram for Admin:



5.2 Data flow diagram for Teacher:



5.3 Data flow diagram for Parent:



6. ER diagram:

Entity relationship diagram displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases.

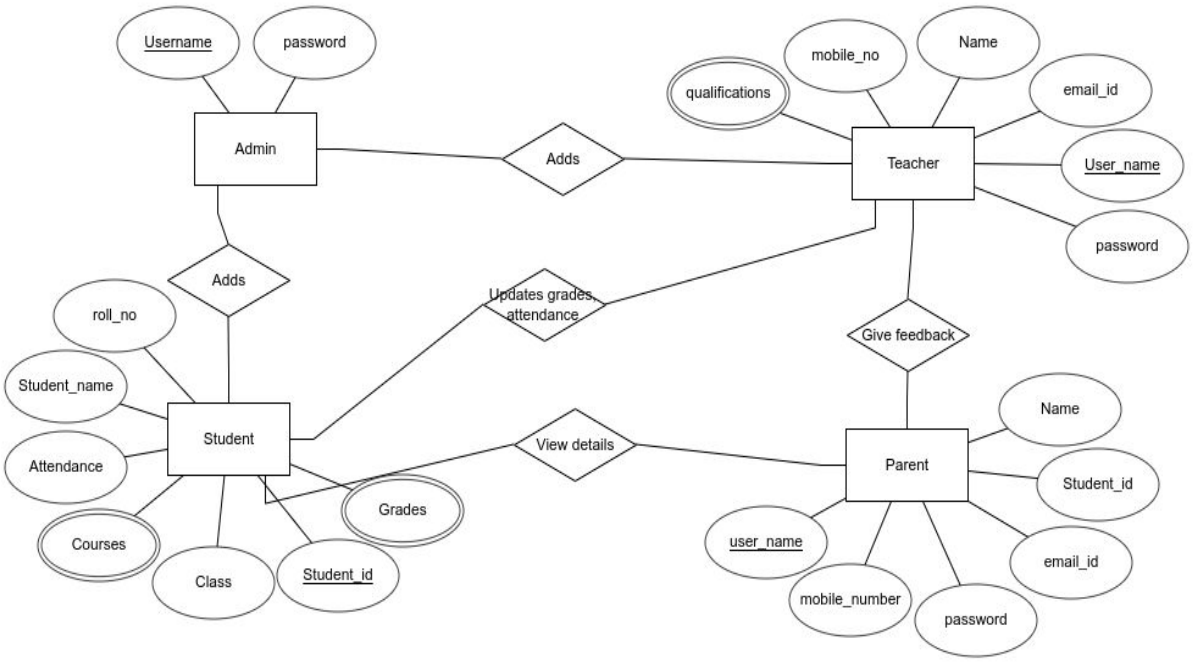
This model is based on three basic concepts:

- Entities
- Attributes
- Relationships

An entity can be a place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some 'attributes' which represent that entity.

It is a single-valued property of either an entity-type or a relationship-type. For example, a lecture might have attributes: time, date, duration, place, etc. An attribute is represented by an Ellipse
A relationship is nothing but an association among two or more entities. Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

Fig6. ER- diagram



7. Navigation diagram:

The purpose of a navigation diagram is to illustrate the navigation on a home page e.g. The individual pages are boxes, with the file name. There's an arrow from one box to another if there's a link from one to the other - so, in this example, there's a link on the page index.html to rules.html. You can also illustrate contact to a database - in this case newgame.asp contacts a database, before producing the ready page, shown in the browser. As with all other kinds of documentation, it should be considered whether a navigation diagram actually makes things clearer. With a very complex diagram, it might be better to have more than one diagram, and then loosely show, what their connection is. In some cases, there are things hard to illustrate - e.g. frames. Some things are easier to write in text - e.g. "and all pages link back to index.html".

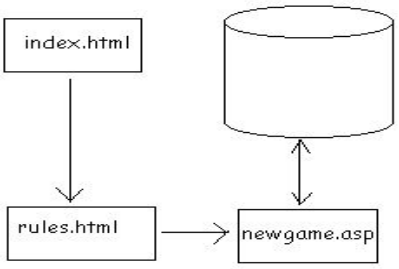
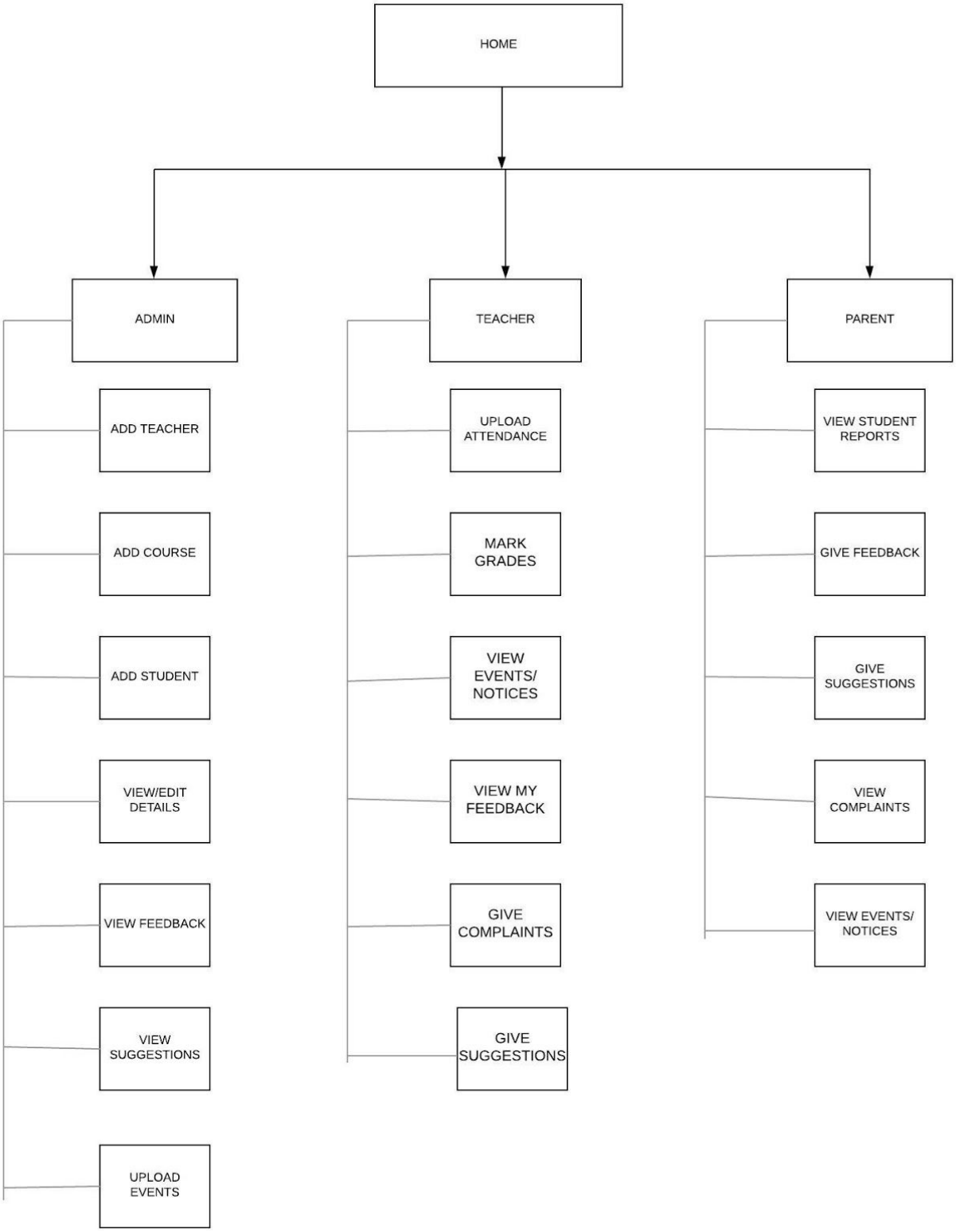


Fig7. Navigation Diagram



8. Component Diagram

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as −

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

Fig8. Component Diagram