**PROJECT**

**ON**

**CENTRAL HIGH SCHOOL SYSTEM**

By

Siva Kumar Garapati

700635222

**Introduction**

The Project on School Management Systems website named as Central High School. In this Web site provide communication between the student, professor and authority’s.

The increasingly Educational Institutions are facing challenges in the organized school system. The Communication medium between people is via the Central High School website. The Website is a dynamic platform which allows educational institutions to manage their whole organization through the web, anytime from anywhere in the world.

This website has the interaction between school, teachers, students and parents in a most effective way. It uses an N-tier architecture developed in Microsoft visual studio 2015, and the back-end service is based around Microsoft SQL Server with some business logic handled by a custom .NET Framework module. The client application is also built with.NET, and handles almost all of the data manipulation and reporting workload. We used Base64 technique to encode and decode of the password. The third party application code uses for the encryption and decryption of the password before saving it to the database.

The purpose of any new technology is to make people life easier. This project is a database used to manage the school and allows the administrators to register the daily required information of Students, Teachers and administrator.

**Technologies used:**

**Development Environment (IDE):** Visual Studio2012

**Database platform:** SQL Server 2014.

**Server-Side Technologies:** Asp.net MVC, C#. Net

**Client-Side Technologies:** HTML5, CSS, JavaScript, JQuery, Bootstrap.

# IMPLIMENTATION

The Central High School Web Site consist of task, such as registering students, record keeping, controlling the data in the database. It produces the different report to the teacher and professors.

In the current system we need to keep a number of records related to the student and want to enter the details of the student and the marks manually. In this System, only the teacher or authority admin views the marks of the student and they want to enter the details of the students.

This System will organize work inside the school and proposed system will do the following tasks:

* Creating the New User’s by Admin
* Insert student's information such as student name, Phone number, address etc.
* Insert Professor's information such as name, number, address, department etc.
* Insert announcement information such as update etc.
* Insert and upload the assignments.
* Insert marks for each student in each subject he/she token.
* View data about certain student and can edit it like adding or removing a student.
* View data about certain employee and can edit it like adding or removing on the employee.
* Insert discussion between the student and professor.

**UML Diagrams:**

UML stands for Unified Modelling Language. Which is used in object-oriented software engineering. It's typically used in software engineering it is a rich language that can be used to model an application structure, behavior and even business processes. There are **14 UML diagram types**.

They can be divided into two main categories are structured diagrams and Behavioral diagrams. The scenarios are organized hierarchically and they capture the system functionality at various abstraction levels, including scenario groups, scenarios, and sub-scenarios. Combining scenarios or sub-scenarios can form complex scenarios. Data are also separately identified, organized, and attached to scenarios. This scenario model can be used to cross check with the UML model. The modeling the logic captured by a single use case or usage scenario, or for modeling the detailed logic of a business rule. Graphical notations used in structural things are the most widely used in UML. The main types of UML Diagrams in this project:

Use case diagram

Sequence diagram

Activity diagram

Class diagram

**Use Case diagram:**

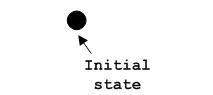
Use case diagrams are a set of use cases, actors and their relationships. They represent the use case view of a system. It represents a particular functionality of a system.

The structure of a system by showing the system's classes, their attributes, operations and the relationships among objects.

The base notation of all the diagrams:

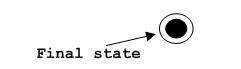
**Initial node**. The filled in circle is the starting point of the diagram.

The initial state is defined to show the start of a process. This notation is used in almost all diagrams.



The usage of Initial State Notation is to show the starting point of a process.

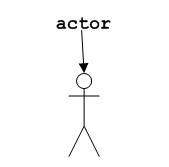
**Activity final node**. The filled circle with a border is the ending point. An activity diagram can have zero or more activity final nodes. This notation is also used in almost all diagrams to describe the end.



The usage of Final State Notation is to show the termination point of a process.

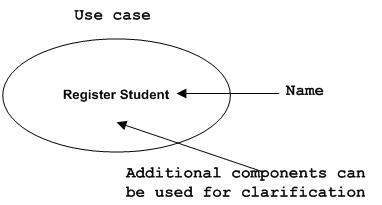
Actor Notation:

An actor can be defined as some internal or external entity that interacts with the system.



Actor is used in a use case diagram to describe the internal or external entities.

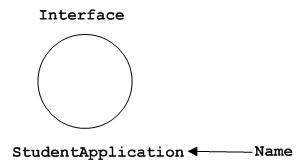
Use case is represented as an eclipse with a name inside it. It may contain additional responsibilities.



Use case is used to capture high-level functionalities of a system.

**Interface Notation**:

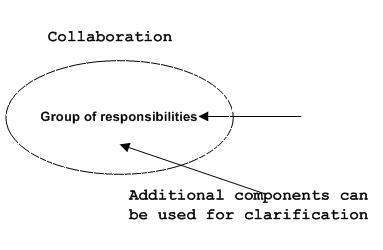
The interface is represented by a circle as shown below. It has a name which is generally written below the circle.



The interface is used to describe functionality without implementation. The interface has been the just like a template where you define different functions not the implementation. When a class implements the interface, it also implements the functionality as per the requirement.

**Collaboration Notation:**

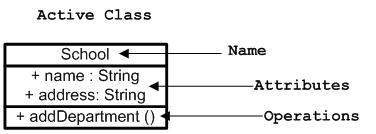
Collaboration is represented by a dotted Eclipse as shown below. It has a name written inside the eclipse.



Collaboration represents responsibilities. Generally responsibilities are in a group.

Active class Notation:

Active class looks similar to a class with a solid border. Active class is generally used to describe the concurrent behavior of a system.

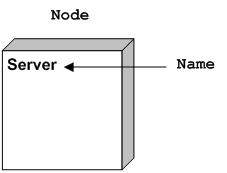


Active class is used to represent concurrency in a system.

**Activity**. The rounded rectangles represent activities that occur. An activity may be physical, such as Inspect Forms, or electronic, such as Display Create Student Screen.

**Join**: A black bar with several flows entering it and one leaving it. All flows going into the joint must reach it before processing may continue. This denotes the end of parallel processing.

**Node Notation:** A node in UML is represented by a square box as shown below with a name. A node represents a physical component of the system.

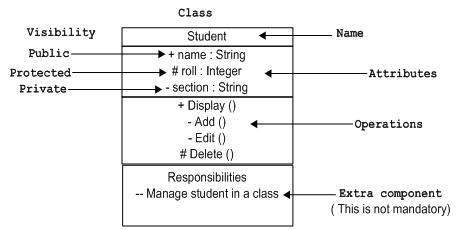


The node is used to represent physical part of a system like server, network etc.

Class Notation:

Class is represented by the diagram shown below. The diagram is divided into four parts.

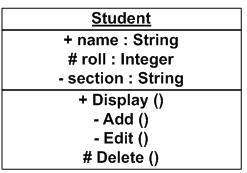
* The top section is used to name the class.
* The second one is used to show the attributes of the class.
* The third section is used to describe the operations performed by the class.
* The fourth section is optional to show any additional components.



Classes are used to represent objects. Objects can be anything having properties and responsibility.

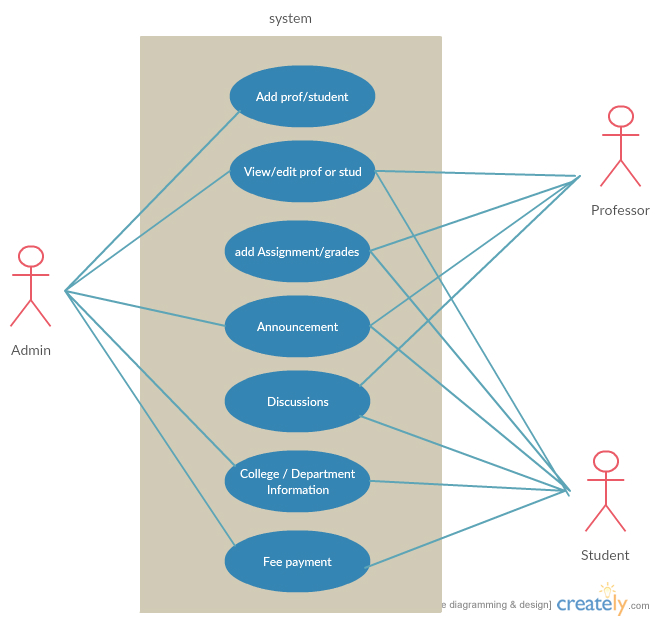
Object Notation:

The object is represented in the same way as the class. The only difference is the name which is underlined as shown below.



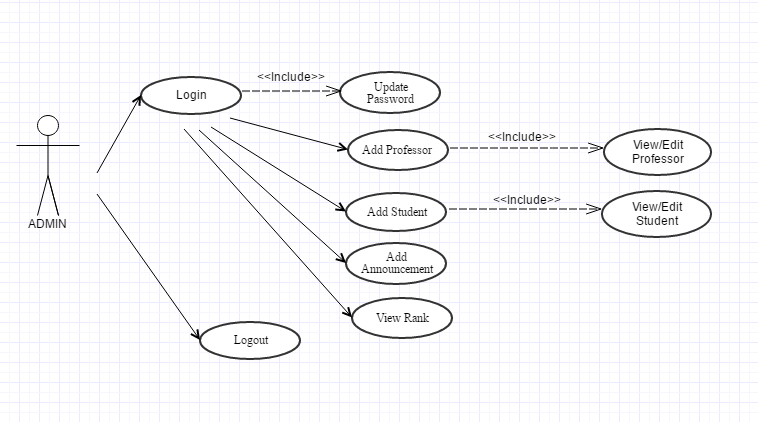
As object is the actual implementation of a class which is known as the instance of a class. So it has the same usage as the class.

Base Use case diagram of the system:

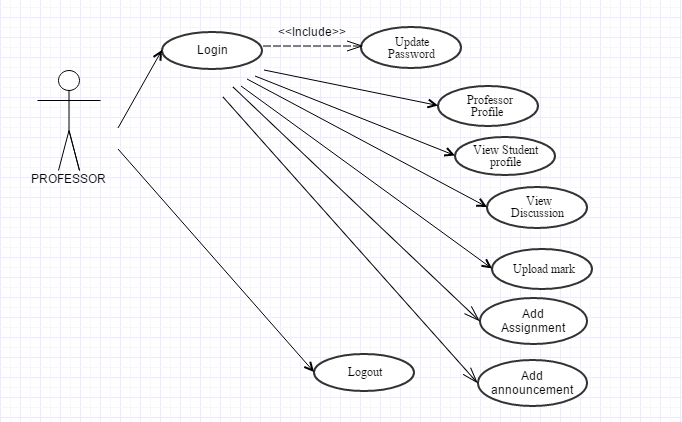


The system has three role for Admin, Professor and Student use case diagram are drawn detail individually below.

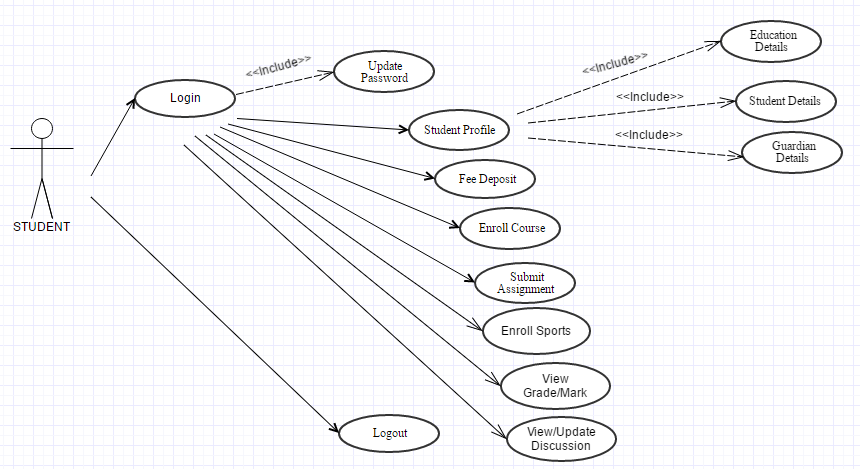
Admin use case:



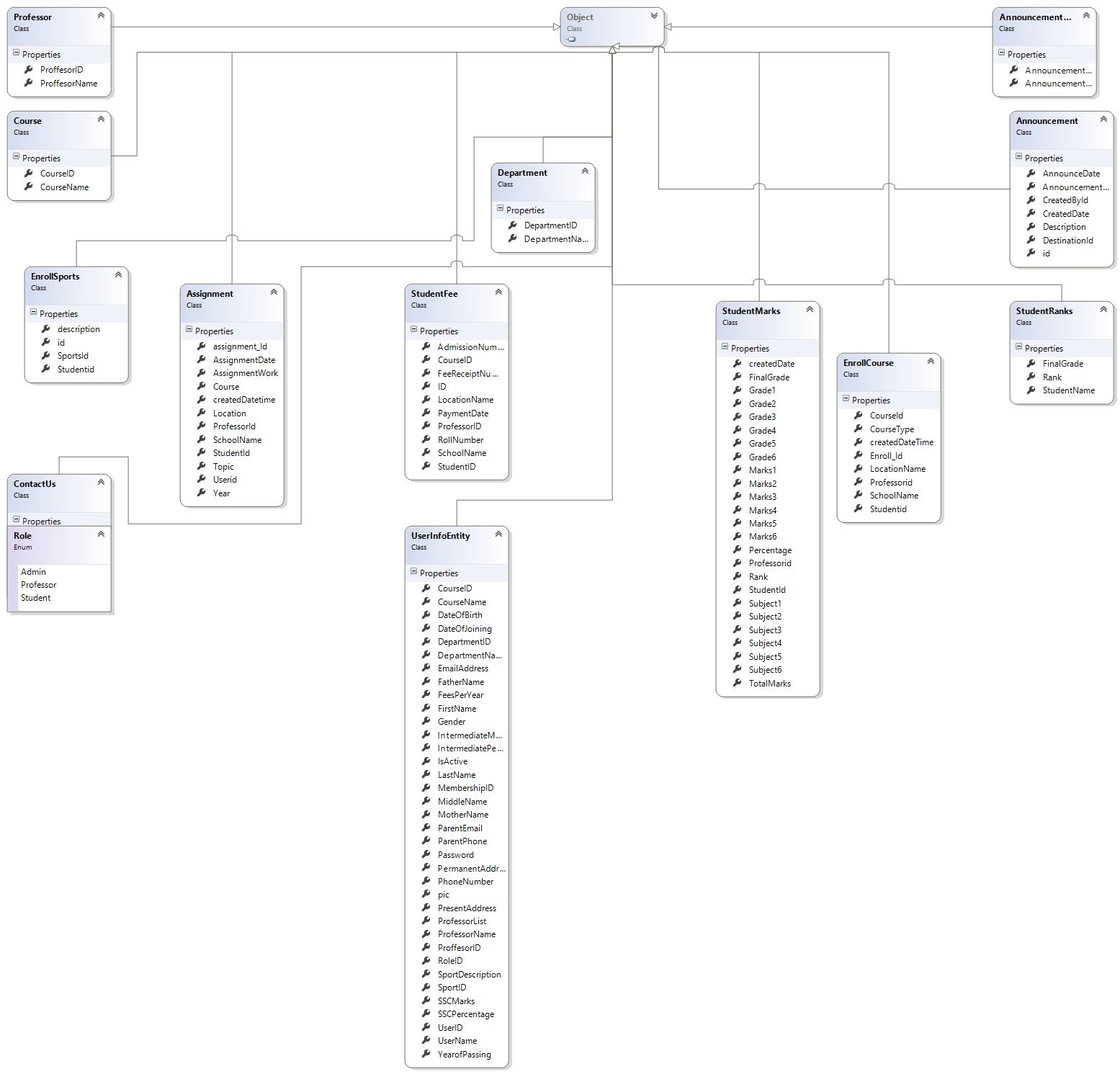
Professor Use case diagram:



**Student Use case Diagram:**



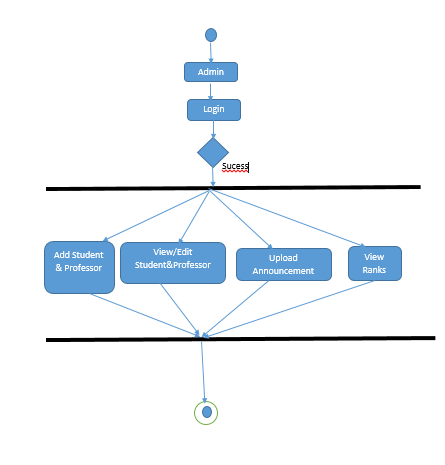
**Class diagrams of the entities:**



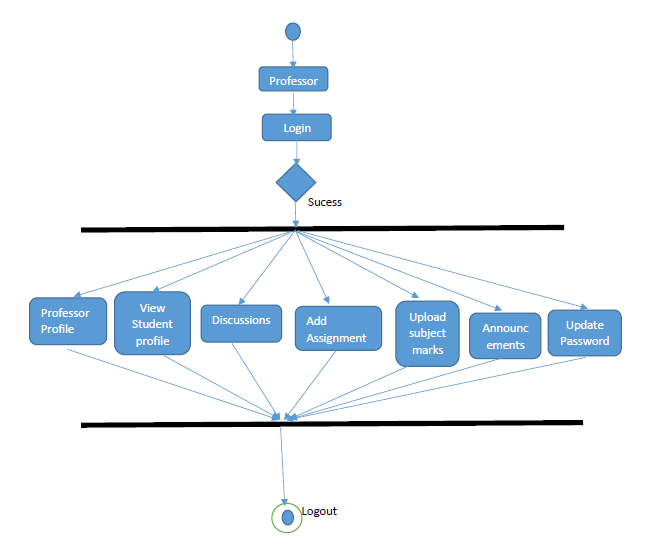
**Activity diagram** describes the business and operational step-by-step activities of the components in a system.

The Admin, Professor and Student activity are step-by-step in below

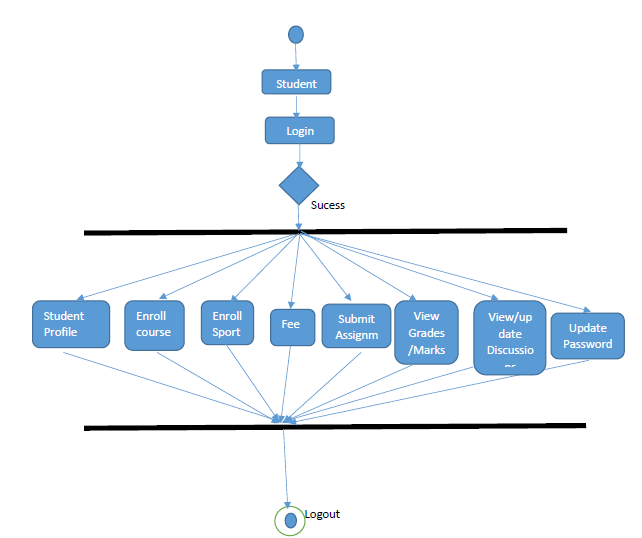
**Admin Activity diagram**



Professor activity diagram



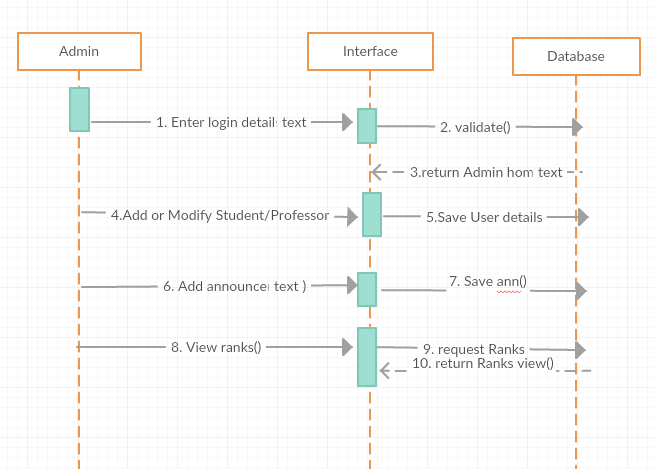
Student Activity diagram:



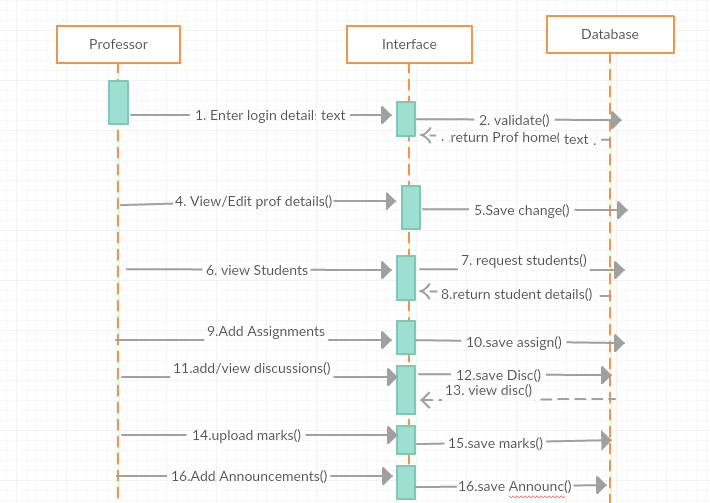
**Sequence diagram:**

The [sequence diagram](https://en.wikipedia.org/wiki/Sequence_diagram) shows how objects communicate with each other in terms of a sequence of messages. Interaction diagrams, a subset of behavior diagrams, emphasize the flow of control and data among the things in the system being modeled.

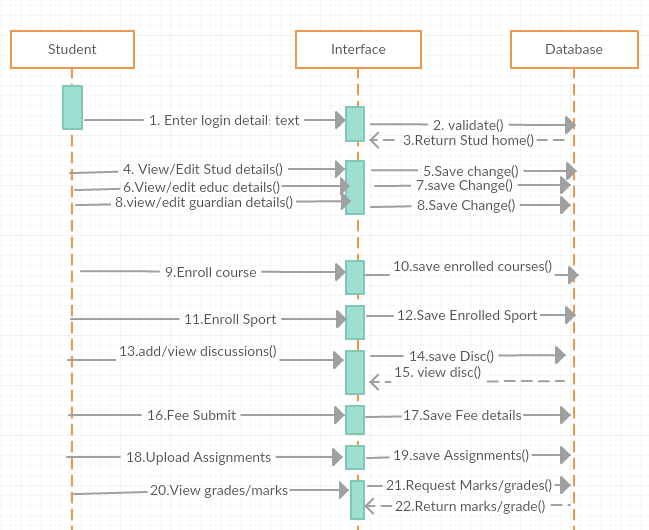
Admin sequence diagram:



Professor sequence diagram:



**Student Sequence diagram:**



**E-R MODEL DIAGRAMS**

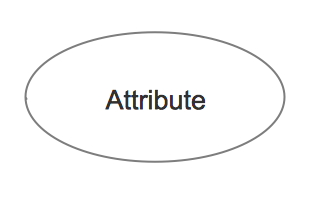
An entity–relationship model (ER model) describes composed of entity types and specifies relationships that can exist between instances of those entity types. The Semantic modeling is modeling data structures, based on the meaning of these data. Different variants of the entity-relationship diagrams are used as a tool for the semantic modeling. ER-model based diagrams have three main components: an entity, a relation and attributes. An entity is a class of similar objects, information about which should be taken into account in the model.

An entity–relationship model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business. It is usually drawn in a graphical form as boxes (entities) that are connected by lines. An ER model is typically implemented as a database.

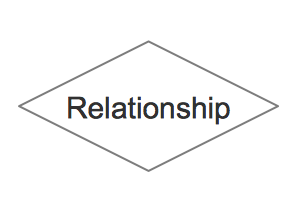
**Entity**: An entity is represented by a rectangle which contains the entity’s name.



**Attribute:** In the Chen notation, each attribute is represented by an oval containing an attribute's name



**Relationship:** A relationship where the entity is existence independent of another entity. A strong relationship is represented by a single rhombus.



In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity.

**Primary key** https://upload.wikimedia.org/wikipedia/commons/5/52/Erd-id-as-primary-key.png

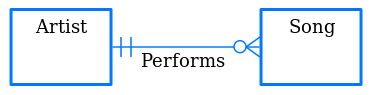
**A relationship with an attribute** https://upload.wikimedia.org/wikipedia/commons/8/84/Erd-relationship-with-attribute.png

**An entity with an attribute** https://upload.wikimedia.org/wikipedia/commons/thumb/9/90/Erd-entity-with-attribute.svg/193px-Erd-entity-with-attribute.svg.png

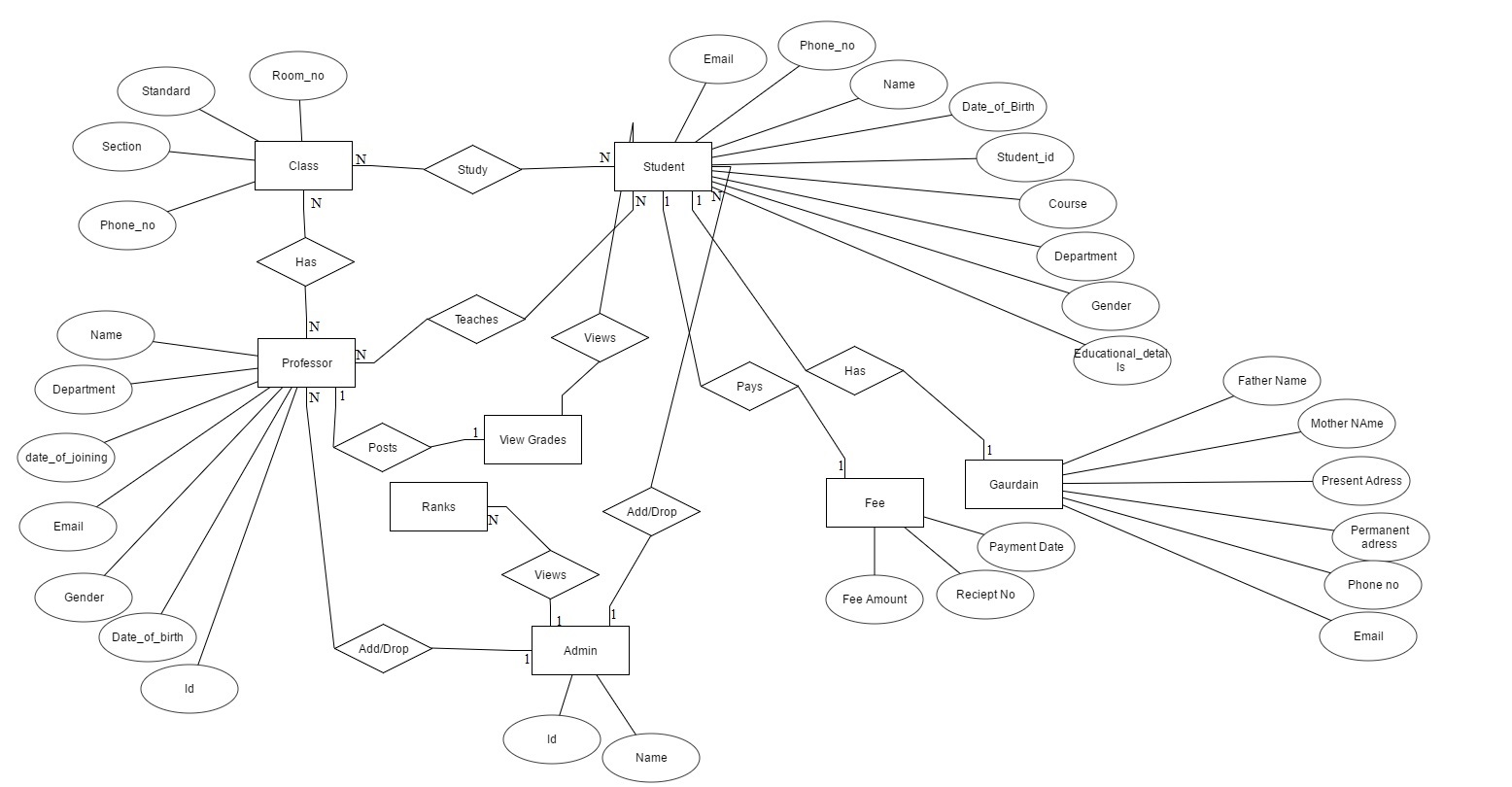
**Two related entities** https://upload.wikimedia.org/wikipedia/commons/thumb/3/3d/Erd-entity-relationship-example1.svg/283px-Erd-entity-relationship-example1.svg.png

**Crow's foot notation**

The Crow's foot diagrams represent entities as boxes, and relationships as lines between the boxes. Different shapes at the ends of these lines represent the entity relationship. In this example, an optional relationship is shown between Artist and Song; the symbols closer to the song entity represent "zero, one, or many", whereas a song has "one and only one" Artist. The former is therefore read as, an Artist (can) perform(s) "zero, one, or many" song(s).



**E-R diagram of this Project**

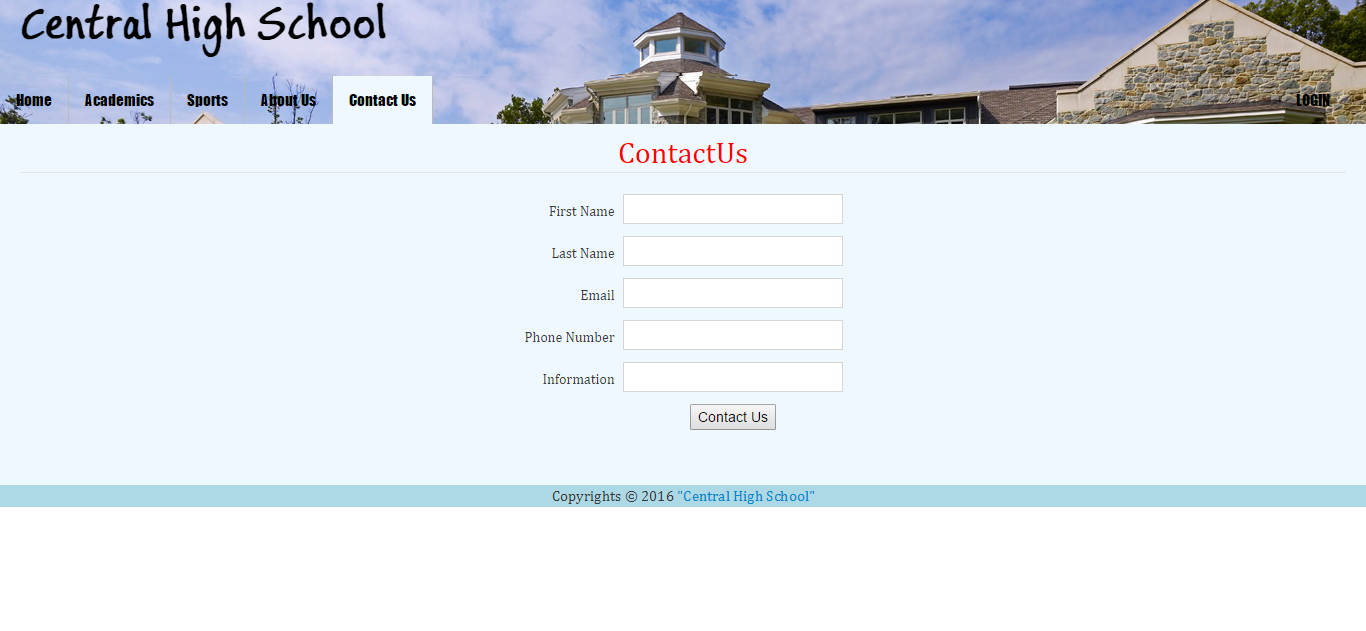


The Screenshot of the Website:

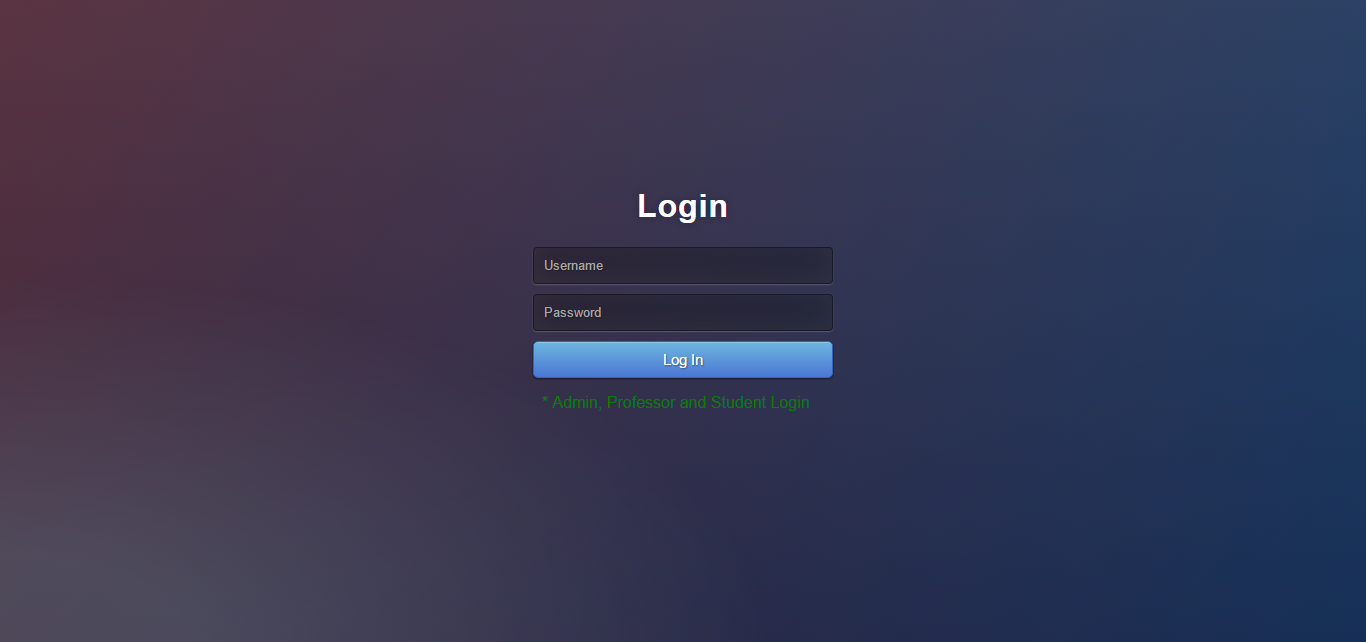
Homepage:



Contact Us page:



Login page:



User home page:

