**AIM:** To draw use case diagram for school management system.

**THEOR**Y: The purpose of a use case diagram in UML is to demonstrate the different ways that a user might interact with a system. A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

* Scenarios in which your system or application interacts with people, organizations, or external systems
* Goals that your system or application helps those entities (known as actors) achieve
* The scope of your system

**When to apply use case diagrams**

A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UML is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

UML use case diagrams are ideal for:

* Representing the goals of system-user interactions
* Defining and organizing functional requirements in a system
* Specifying the context and requirements of a system
* Modeling the basic flow of events in a use case

**DESCRIPTION:**

* **Admin:**

The admin keeps the record of:-

* Students getting enrolled
* Teachers getting enrolled
* Attendance of teachers
* Evaluation of performance of teachers
* **Teacher:**

The teacher keeps the following records:

* Student attendance
* Student academic record
* Evaluate student performance
* Generate reports and notices
* **Principal:**

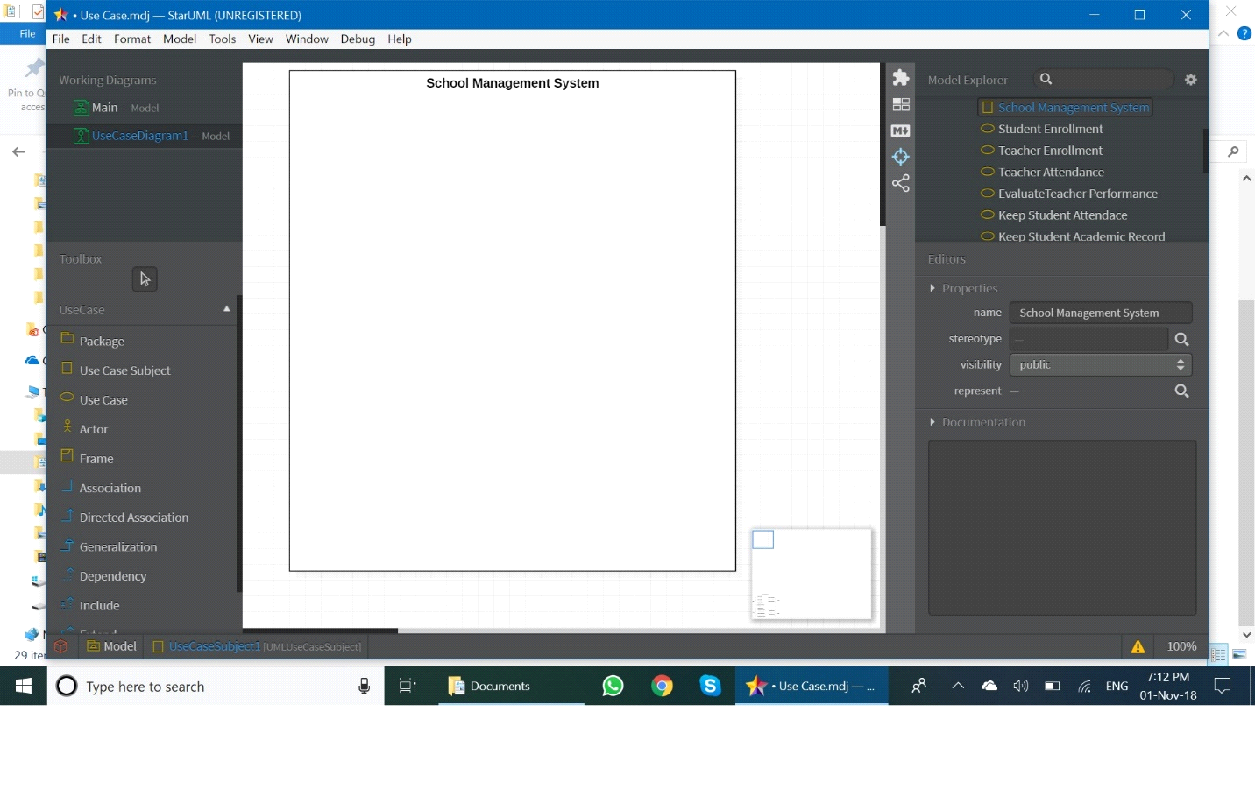
Roles of a principal:

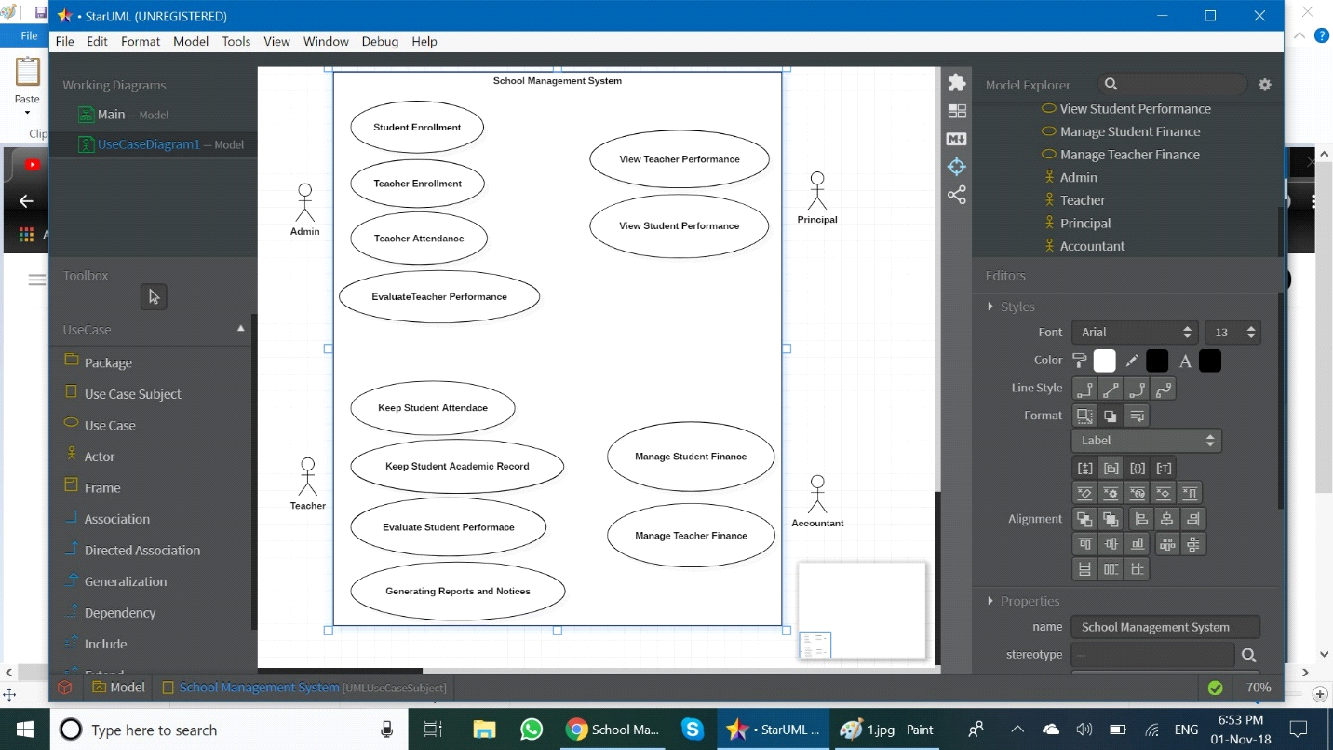
* View teacher performance
* View student performance
* **Accountant:**

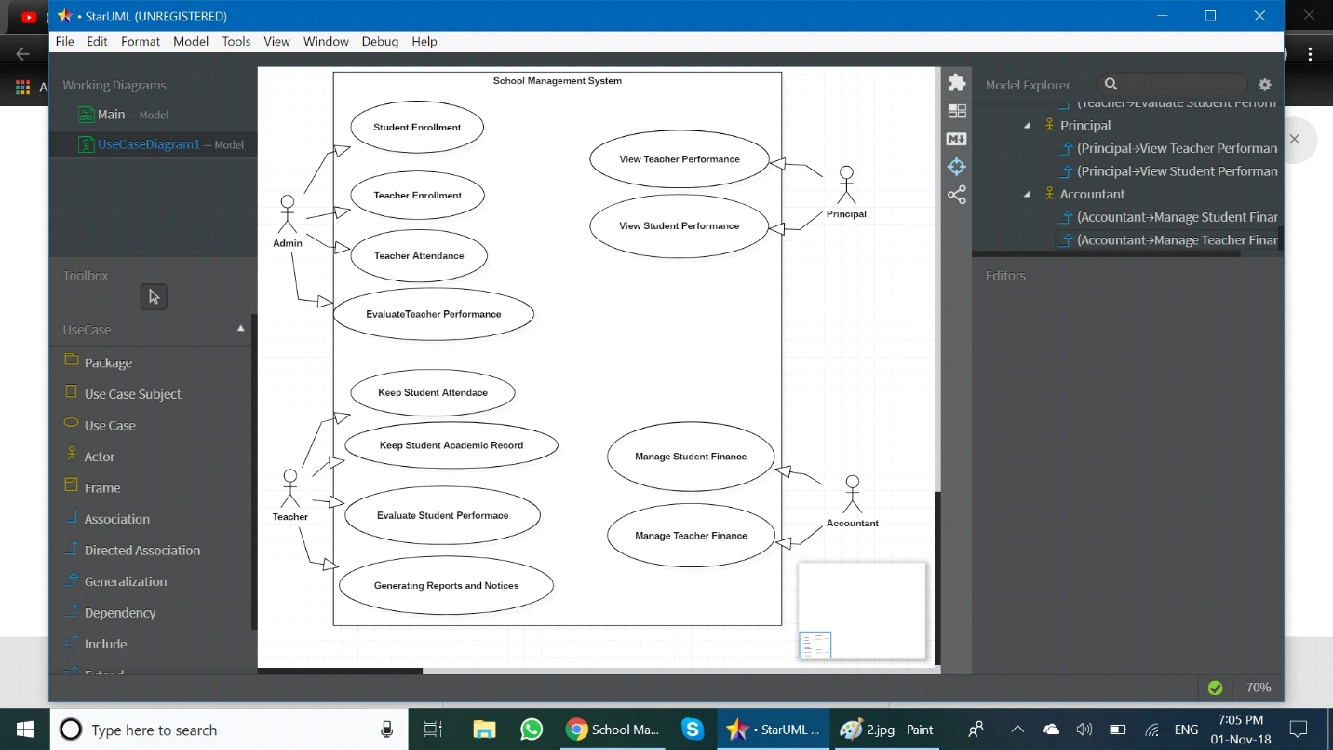
Role of an accountant:

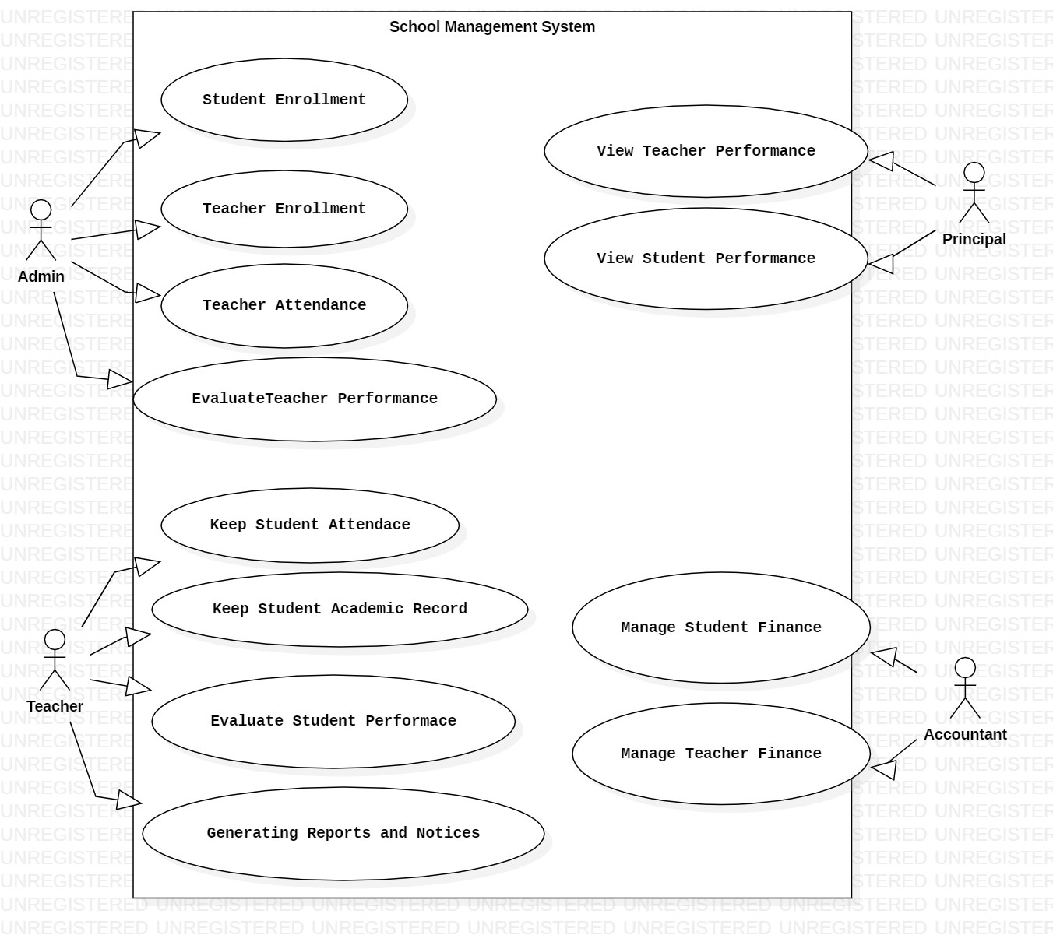
* Manage student finance
* Manage teacher finance

**DIAGRAM:**









**AIM :** To draw the class diagram for School Management System

**THEORY :** A class diagram is a type of static structure diagram that describes the structure

of a system by showing the system classes, their attributes, and the relationships between the classes.

Class diagrams show the classes of the system, their inter-relationships, and the operations

and attributes of the classes. Class diagrams are typically used, although not all at once, to:

* Explore domain concepts in the form of a domain model
* Analyze requirements in the form of a conceptual/analysis model
* Depict the detailed design of object-oriented or object-based software

A class model is comprised of one or more class diagrams and the supporting specifications

that describe model elements including classes, relationships between classes, and interfaces.

There are guidelines

* General issues
* Classes
* Interfaces
* Relationships
* Inheritance
* Aggregation and Composition

**GENERAL GUIDELINES**

Because class diagrams are used for a variety of purposes – from understanding requirements

to describing your detailed design – it is needed to apply a different style in each

circumstance. This section describes style guidelines pertaining to different types of class

Diagrams.

**CLASSES:** A class in the software system is represented by a box with the name of the class

written inside it. A compartment below the class name can show the class&#39;s attributes (i.e. its properties). Each attribute is shown with at least its name, and optionally with its type, initial

value, and other properties. A class is effectively a template from which objects are created

(instantiated). Classes define attributes, information that is pertinent to their instances, and

operations, functionality that the objects support. Classes will also realize interfaces (more on

this later). Class diagrams are widely used to describe the types of objects in a system and

their relationships. Class diagrams model class structure and contents using design elements

such as classes, packages and objects. Class diagrams describe three different perspectives

when designing a system, conceptual, specification, and implementation. These perspectives

become evident as the diagram is created and help solidify the design.

**INTERFACES:** An interface is a collection of operation signature and/or attribute definitions

that ideally defines a cohesive set of behaviors. Interface a class or component must

implement the operations and attributes defined by the interface. Any given class or

component may implement zero or more interfaces and one or more classes or components

can implement the same interface.

**RELATIONSHIPS:** A relationship is a general term covering the specific types of logical

connections found on a class and object diagram. Class diagrams also display relationships

such as containment, inheritance, associations and others. The association relationship is the

most common relationship in a class diagram. The association shows the relationship

between instances of classes. Another common relationship in class diagrams is a

generalization. A generalization is used when two classes are similar, but have some

Differences.

**AGGREGATION:** Aggregation is a variant of the &quot;has a&quot; or association relationship; composition is more specific than aggregation. Aggregation occurs when a class is a collection or container of other classes, but where the contained classes do not have a strong

life cycle dependency on the container--essentially, if the container is destroyed, its contents

are not.

**ASSOCIATION:** Association are semantic connection between classes. When an association

connects two classes , each class can send messages to other in a sequence or a collaboration

diagram . Associations can be bidirectional or unidirectional.

**DEPENDENCIES:** Dependencies connect two clases . Dependencies are always

unidirectional and show that one class, depends on the definitions in another class .

**GENERALIZATION:** The generalization relationship indicates that one of the two related

classes (the supertype) is considered to be a more general form of the other (the subtype). In

practice, this means that any instance of the subtype is also an instance of the supertype . The

generalization relationship is also known as the inheritance or &quot;is a&quot; relationship. The

supertype in the generalization relationship is also known as the &quot;parent&quot;, superclass, base

class, or base type. The subtype in the generalization relationship is also known as the

&quot;child&quot;, subclass, derived class, derived type, inheriting class, or inheriting type.

**MULTIPLICITY:** The association relationship indicates that (at least) one of the two related

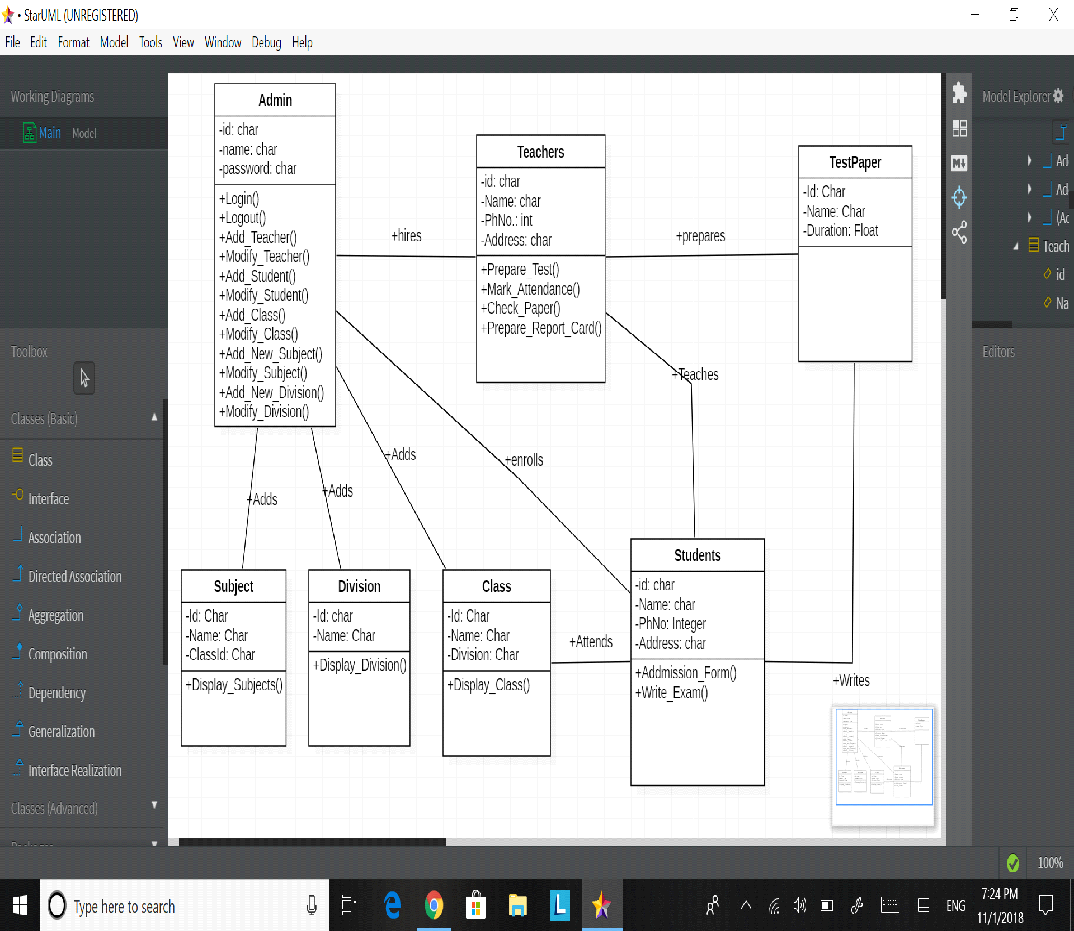
classes makes reference to the other.

**Description**

* **Teacher**
* **Introduction:** The class Teacher represents all the attributes or information of
* the teacher that must be known by the school and all the operations performed by the teacher. It has its associativity with Admin, Students and TestPaper.
* **Attributes**
* Name: It contains the name of the teacher.
* Id: It contains the unique id that has been given to the teacher.
* PhNo.: It contains the Phone no of the teacher.
* Address: It contains the latest residential address of the teacher.
* **Operations**
* Prepare\_Test\_Papers(): The teacher prepars the test papers to be given to the students for assessments.
* Mark\_Attendence(): Teachers marks the attendance of the students present in the class daily
* Check\_Papers(): Teachers checks the papers that were given by the students.
* Prepare\_Report\_Cards(): teacher has to compile the result of all the test for each student and save them
* Declare\_Result(): teachers need to declare the result of all the students.
* **Student**
* **Introduction:** The class student represents the children who comes to the school for the studying. It is associated to teacher, testpaper and class
* **Attributes**
* Name: It contains the name of the student.
* PhNo: It contains the phone number of the student.
* Address: It contains the address of the student
* Id: It contains the id of the student, a unique number given to the student.
* **Operations**
* FIll\_Admission\_Form(): Student need to fill this before getting admitted to the school
* Get\_enrolled(): Student need to get themselves enrolled to the school before attending the shcool
* Write\_exam(): Student need to write there exams to get promoted to a new class.
* **Test Paper**
* **Introduction:** Students need to attempt the papers made by the teacher so as to promote to the next division in school.
* **Attributes**
* ID: It contains Unique id of the testpaper
* Name: It contains Name of the exam
* Duration: it contains duration of the exam
* Classid: It contains the unique class id for which the exam is to be conducted
* **ADMIN**
* **Introduction:** The Admin class is handled at the reception of the School , to

manage all the activities of the school - to provide Add and modify teachers information ; for login and logout ; to add and modify Students information; to Add and modify class information; to add and modify division information

* **Attributes**
* ID: It contains the Unique Id No. of the Admin
* Name:It contains Name of the admin
* Password: It contains the unique password of a particular admin
* **Operations**
* Login(): this helps admin to log in to the portal
* Logout(): this helps admin to log out of the portal
* Add\_New\_Teachers(): admin can add new teacher information using this
* Modify\_Teachers(): admin can modify teacher information using this.
* Add\_New\_Student(): admin can add new student information using this
* Modify\_Student(): admin can modify student information using this
* Add\_New\_Subject(): admin can add new subject using this
* Modify\_Subject(): admin can modify current subjects using this
* Add\_New\_Class(): admin can add new class using this
* Modify\_Class(): admin can modify already formed class using this
* Add\_New\_Division(): admin can add new division to the school using this
* Modify\_Division(): admin can modify current divisions of the school using this
* **Subject**
* **Introduction:** These are the current subjects that are taught to a particular class. These can be changed or modified only by the admin.
* **Attributes:**
* ID: This contains the unique subject ID
* Name: This contains the name of the subject
* ClassID: This contains the ID of the class the subject is taught to
* **Operations**
* Display\_Subjects(): used to display the information about all the subjects
* **Division**
* **Introduction:** These are the different division present in the school. These can only be changed by the Admin.
* **Attributes:**
* ID: it contains the unique division id for each division
* Name: It contains the name of the division
* **Operations:**
* Display\_Divisions(): This helps in displaying all the information about all the divisions.
* **Class**
* **Introduction:** There are various classes in a school. These are contained in this block and can be modified only by the admin. Each class contains Students that attend their classes and teachers who teaches these students.
* **Attributes:**
* ID: contains unique id of each class.
* Name: contains the name of the class or else known as section.
* Division: contains the information about which division the class belongs to.
* **Operations:**
* Display\_Class(): It displays all the attributes of class block.



**Objective:** To create state transition diagram for School Management System

**Theory:**

State-transition diagrams describe all the states that an object can have, the events under which an object changes state (transitions), the conditions that must be fulfilled before the transition will occur (guards), and the activities undertaken during the life of an object (actions). State-transition diagrams are very useful for describing the behaviour of individual objects over the full set of use cases that affect those objects. State-transition diagrams are not useful for describing the collaboration between objects that cause the transitions.

**Description**

* **States**

States represent situations during the life of an object. You can easily illustrate a

state in SmartDraw by using a rectangle with rounded corners.

* **Transition**

A solid arrow represents the path between different states of an object. Label

the transition with the event that triggered it and the action that results from it.

A state can have a transition that points back to itself.

* **Initial State**

A filled circle followed by an arrow represents the object&#39;s initial state.

* **Final State**

An arrow pointing to a filled circle nested inside another circle represents the

object&#39;s final state.

* **Synchronization and Splitting of Control**

A short heavy bar with two transitions entering it represents a synchronization

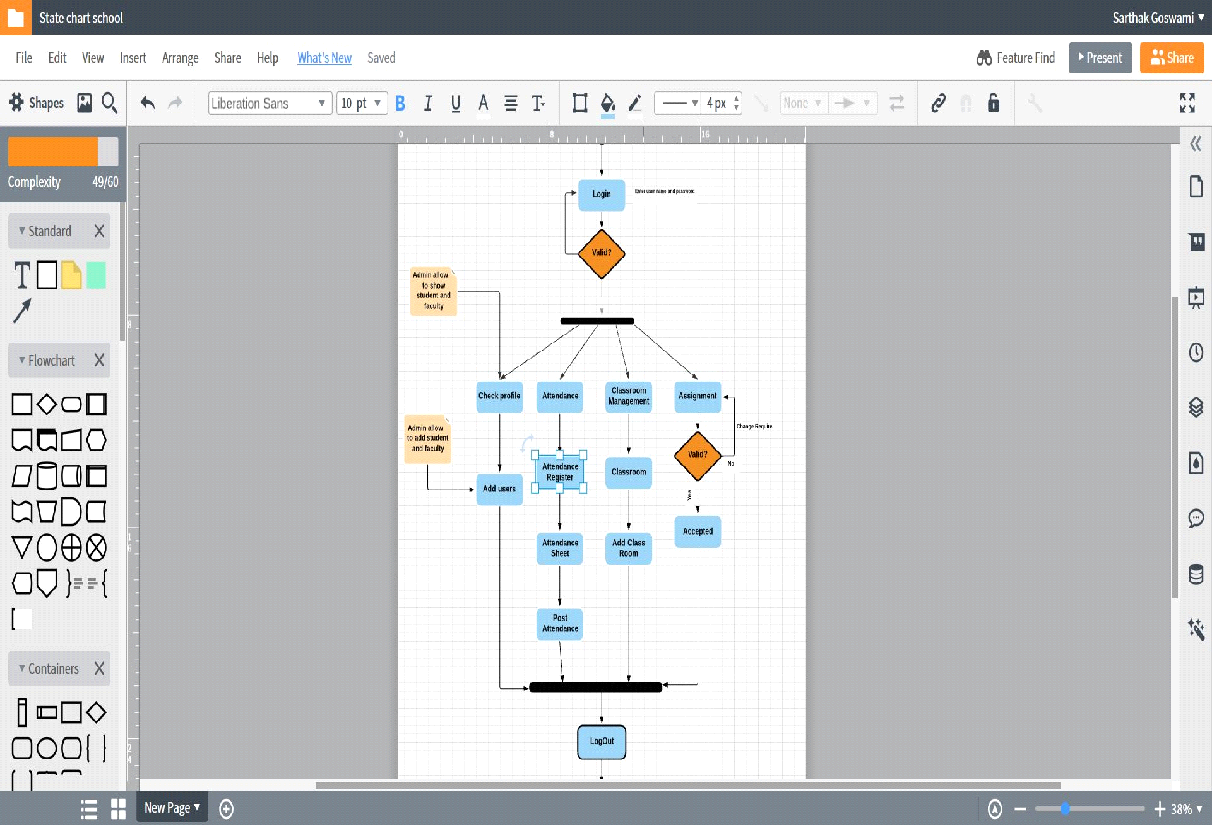
of control. The first bar is often called a fork where a single transition splits into

* **Concurrent multiple transitions**.

The second bar is called a join, where the concurrent transitions reduce back to one.

**States-**

* **Teacher**
* **Introduction:** The class Teacher represents all the attributes or information of the teacher that must be known by the school and all the operations performed by the teacher. It has its associativity with Admin, Students and TestPaper.
* **Attributes**
* Name: It contains the name of the teacher.
* Id: It contains the unique id that has been given to the teacher.
* PhNo.: It contains the Phone no of the teacher.
* Address: It contains the latest residential address of the teacher.
* **Operations**
* Prepare\_Test\_Papers(): The teacher prepars the test papers to be given to the students for assessments.
* Mark\_Attendence(): Teachers marks the attendance of the students present in the class daily
* Check\_Papers(): Teachers checks the papers that were given by the students.
* Prepare\_Report\_Cards(): teacher has to compile the result of all the test for each student and save them
* Declare\_Result(): teachers need to declare the result of all the students.
* **Student**
* **Introduction:** The class student represents the children who comes to the school for the studying. It is associated to teacher, testpaper and class
* **Attributes**
* Name: It contains the name of the student.
* PhNo: It contains the phone number of the student.
* Address: It contains the address of the student
* Id: It contains the id of the student, a unique number given to the student.
* **Operations**
* FIll\_Admission\_Form(): Student need to fill this before getting admitted to the school
* Get\_enrolled(): Student need to get themselves enrolled to the school before attending the shcool
* Write\_exam(): Student need to write there exams to get promoted to a new class.
* **Test Paper**
* **Introduction:** Students need to attempt the papers made by the teacher so as to promote to the next division in school.
* **Attributes**
* ID: It contains Unique id of the testpaper
* Name: It contains Name of the exam
* Duration: it contains duration of the exam
* Classid: It contains the unique class id for which the exam is to be conducted
* **ADMIN**
* **Introduction:** The Admin class is handled at the reception of the School , to manage all the activities of the school - to provide Add and modify teachers information ; for login and logout ; to add and modify Students information; to Add and modify class information; to add and modify division information
* **Attributes**
* ID: It contains the Unique Id No. of the Admin
* Name:It contains Name of the admin
* Password: It contains the unique password of a particular admin
* **Operations**
* Login(): this helps admin to log in to the portal
* Logout(): this helps admin to log out of the portal
* Add\_New\_Teachers(): admin can add new teacher information using this
* Modify\_Teachers(): admin can modify teacher information using this.
* Add\_New\_Student(): admin can add new student information using this
* Modify\_Student(): admin can modify student information using this
* Add\_New\_Subject(): admin can add new subject using this
* Modify\_Subject(): admin can modify current subjects using this
* Add\_New\_Class(): admin can add new class using this
* Modify\_Class(): admin can modify already formed class using this
* Add\_New\_Division(): admin can add new division to the school using this
* Modify\_Division(): admin can modify current divisions of the school using this
* **Subject**
* **Introduction:** These are the current subjects that are taught to a particular class.These can be changed or modified only by the admin.
* **Attributes:**
* iD: This contains the unique subject ID
* Name: This contains the name of the subject
* ClassID: This contains the ID of the class the subject is taught to
* **Operations**
* Display\_Subjects(): used to display the information about all the subjects
* **Division**
* Introduction: These are the different division present in the school. These can only be changed by the Admin.
* **Attributes:**
* ID: it contains the unique division id for each division
* Name: It contains the name of the division
* **Operations:**
* Display\_Divisions(): This helps in displaying all the information about all the divisions.
* **Class**
* **Introduction:** There are various classes in a school. These are contained in this block and can be modified only by the admin. Each class contains Students that attend their classes and teachers who teaches these students.
* **Attributes:**
* ID: contains unique id of each class.
* Name: contains the name of the class or else known as section.
* Division: contains the information about which division the class belongs to.
* **Operations:**
* Display\_Class(): It displays all the attributes of class block.



**Aim :** To draw a Sequence Diagram of School Management System.

**Software Required :** STARUML

**Theory :** Sequence diagrams are a popular dynamic modeling solution in UML because they specifically focus on *lifelines*, or the processes and objects that live simultaneously, and the messages exchanged between them to perform a function before the lifeline ends. Along with our UML diagramming tool, use this guide to learn everything there is to know about sequence diagrams in UML.

**Description :**

The various functions in the this school management system are as follows

1. PrepareTestPaper() – The exam that the students will write is prepared , which is then displayed by this function.

2 . Add/ModifyStudent() – To add names of new students or make some changes to the existiong ones.

3 . DisplaySubjects() – To display the subjects that the students study.

4 . DisplayDivision() – To display the division that is, the class or section in which students study.

5 . DisplayClass() – To display the class of the students

6. HireTeacher() – To display the hired teacher in school

The rest functions such as modifyclass() , modifysubjects() , modifydivision() are for doing any changes in the management system of the particular school.

7 . WriteExam() – The exam that is to be written by students, prepared by teachers is dispayed.

8 . MarkAttendance() – The attendance is marked.

9 . PrepareReportCard() – The report card is prepared that shows the marks and performance of students.

Return() is used to call the respective functions

**Diagram :**

