**3.1 Introduction to System Design**

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

**3.2 Proposed System and its Features**

This system can be used for managing school resource to get the maximum utilization of them. The administrator logs in the system and should be able to register new student, register new employee and modify student record and result. The administrator also gets view of stored record, fee, student data, staff salary and other records.

The main purpose of the project is to computerize the manual system and reduce the time consumption and paper work. Some objectives are

* Make all the system computerized
* Reduce the time consumption
* All the system managements automated
* Reduce error scope
* Less paper work
* Centralized data management system
* Easy operations for operator of the system

**3.3 System Design using UML**

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, 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.1 The UML is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

**Goals of UML**

The primary goals in the design of the UML were:

* Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
* Provide extensibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development processes.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of the OO tools market.
* Support higher-level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

**Why Use UML?**

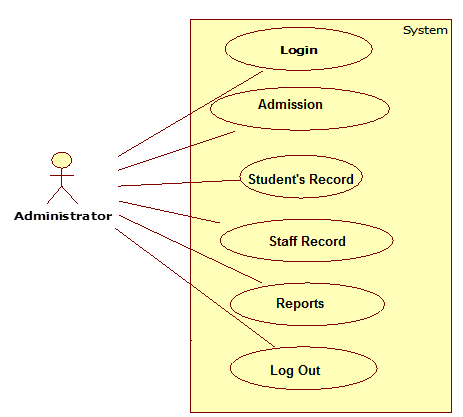
As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modeling Language (UML) was designed to respond to these needs.

Each UML diagram is designed to let developers and customers view a software system from a different perspective and in varying degrees of abstraction. UML diagrams commonly created in visual modeling tools include.

Here is now use case diagram for the school management System which can represents Admin and user (faculty member).

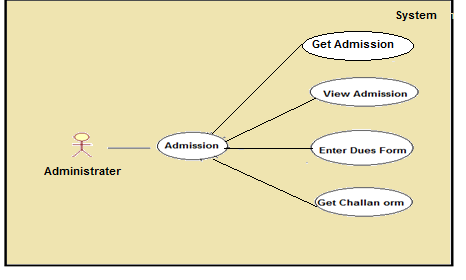
**3.3.1 Use Case Diagrams**

**Full System Use Case View**

****

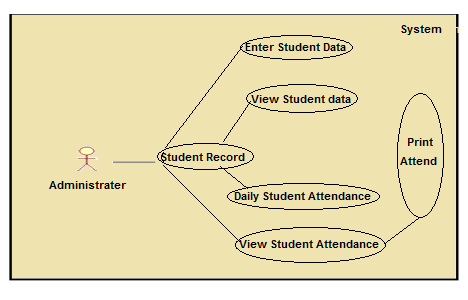
**Figure 3.1 full system views**

**Admission Use Case View**

****

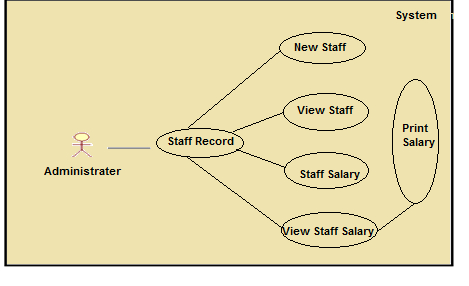
**Figure 3.2 admission system views**

**Student Record Use Case View**

****

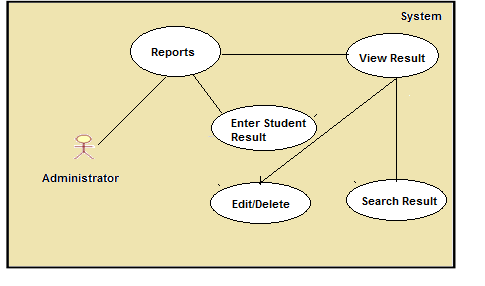
**Figure 3.3 student record system views**

**Staff Record Use Case View**

****

**Figure 3.4 staff record system views**

**Result/Reports Use Case View**

****

**Figure 3.5 result/reports system views**

It is easy to mix up the definitions of use case and use case scenario. A use case represents the actions that are required to enable or abandon a goal. A use case has multiple “paths” that can be taken by any user at any one time. A use case scenario is a single path through the use case. This article provides an example use case and some diagrams to help visualize the concept.

A usage scenario, or scenario for short, describes a real-world example of how one or more people or organizations interact with a system.  They describe the steps, events, and/or actions which occur during the interaction.  Usage scenarios can be very detailed, indicating exactly how someone works with the user interface, or reasonably high-level describing the critical business actions but not the indicating how they’re performed.

Here is some Usage Scenario with UML Activity diagrams which can show the process flow of the School Management System.

**User Case:**

|  |  |
| --- | --- |
| Use Case ID | 01 |
| **Use Case Name** | **Login** |
| Primary Actor | Administrator |
| Brief Description | In this case user administrator enters his username and password to get logged in so that he may use the functionality of system and this process also make user session. An interface is provided to user with text field for putting input or name and password. |
| Precondition | Administrator must have an administrator rights. |
| Basic Flows | 1. Administrator enters his/her id and password. 2. Presses login button. 3. Administrator logged in when the user name and password is correct. |
| Alternative  Flows | User name or password incorrect.  System will display that the user name or password is wrong. |
| Post Condition | After authentication of user name and password, administrator is now able to use the system database according to his/her privilege. |

|  |  |
| --- | --- |
| Use Case ID | 02 |
| **Use Case Name** | **Add (Register), Edit, and Search Student’s Record**. |
| Primary Actor | Administrator. |
| Brief Description | In this use case administrator will have the rights to add , edit and search student. |
| Preconditions | Administrator is already login. |
| Basic Flows | Administrator simply adds, edit and view student’s record. |
| Post Condition | System shows the error message to administrator if those student records already exist in database. |

|  |  |
| --- | --- |
| Use Case ID | 03 |
| **Use Case Name** | **View All Students By Name and Class** |
| Primary Actor | Administrator. |
| Brief Description | In this use case administrator will see all the register students names, roll no, father name and class in one place by class or year or name. |
| Preconditions | Administrator is already login and view student list option is available. |
| Basic Flows | Administrator simply view register student record by class, year or by student name. |
| Post Condition | System shows all the records which is generated by the request. |

|  |  |
| --- | --- |
| Use Case ID | 04 |
| **Use Case Name** | **Add, View Monthly Fees Record.** |
| Primary Actor | Administrator. |
| Brief Description | In this use case administrator will have the rights to add and edit student monthly fees record and the auto fees slip is generated. |
| Preconditions | Administrator is already login. |
| Basic Flows | Administrator simply adds and edits student monthly fees record. |
| Post Condition | System shows the error message to administrator if that monthly fees record is already exists in database. |
| Use Case ID | 05 |
| **Use Case Name** | **View Deposit Fees Slip.** |
| Primary Actor | Administrator. |
| Brief Description | In this use case administrator will see fees slip which is auto generated when the fees record is add and by request that by student roll no. |
| Preconditions | Administrator is already login. |
| Basic Flows | Administrator simply view fees slip. |
| Post Condition | System shows to administrator fees slip auto generated by request that by student roll no. |

|  |  |
| --- | --- |
| Use Case ID | 06 |
| **Use Case Name** | **Enter Student Attendance and View Attendance** |
| Primary Actor | Administrator. |
| Brief Description | In this use case administrator will inter the student’s attendance and view the daily attendance with selected date. |
| Preconditions | Administrator is already login. |
| Basic Flows | Administrators simply view student attendance. |
| Post Condition | System shows to administrator the student attendance that auto generated by date. |

|  |  |
| --- | --- |
| Use Case ID | 07 |
| **Use Case Name** | **View Faculty Members.** |
| Primary Actor | Administrator. |
| Brief Description | In this use case administrator will see all the register faculty member name, faculty id, father name and phone no. in one place. |
| Preconditions | Administrator is already login and view faculty member list option is available. |
| Basic Flows | Administrators simply view all faculty member record. |
| Post Condition | System shows all the records which is generated by the request. |

|  |  |
| --- | --- |
| Use Case ID | 08 |
| **Use Case Name** | **View Faculty Member Record.** |
| Primary Actor | Administrator |
| Brief Description | In this use case administrator will have the rights to view faculty member record. |
| Preconditions | Administrator is already login. |
| Basic Flows | Administrator simply views s faculty member record. |

|  |  |
| --- | --- |
| Use Case ID | 09 |
| **Use Case Name** | **Edit and Delete Student Exam Result.** |
| Primary Actor | Administrator. |
| Brief Description | In this use case administrator will have the rights to add and edit student exam result. |
| Preconditions | Administrator is already login. |
| Basic Flows | Administrator simply adds and edits student exam result. |
| Post Condition | System shows the error message to administrator if the obtain marks is greater than total marks. |

|  |  |
| --- | --- |
| Use Case ID | 10 |
| **Use Case Name** | **View Result.** |
| Primary Actor | Administrator. |
| Brief Description | In this use case administrator will see result card auto generated by request that by student roll no. |
| Preconditions | Administrator is already login and view result card option is available. |
| Basic Flows | Administrator simply view result card. |
| Post Condition | System shows the administrator result card auto generated by request that by student roll no. |

|  |  |
| --- | --- |
| Use Case ID | 11 |
| **Use Case Name** | **AddDelete, and View User Accounts.** |
| Primary Actor | Administrator. |
| Brief Description | In this use case administrator will have the right to add, delete and view user accounts. |
| Preconditions | Administrator is already login. |
| Basic Flows | Administrator simply adds, delete and view user accounts. |

|  |  |  |
| --- | --- | --- |
| Use Case ID | | 12 |
| Use Case Name | | **Change Accounts Password**. |
| Primary Actor | | Administrator. |
| Brief Description | | In this use case administrator will have to login to reset password by putting the user name, correct old, new and confirm new password. This reset password message will be shown in front of Administrator. |
| Preconditions | | Administrator is already login. |
| Basic Flows | | Administrator simply views user information and about us page and all access to the web. |
| Use Case ID | | 13 | | |
| Use Case Name | | **Change Password.** | | |
| Primary Actor | | Administrator. | | |
| Brief Description | | In this use case Administrator will have to login to change password by putting correct old, new and confirm new password. This rest password message will be shown in front of you. | | |
| Preconditions | | Administrator is already login. | | |
| Basic Flows | | Administrator simply views information and about us page and all access to the web. | | |

|  |  |
| --- | --- |
| Use Case ID | 14 |
| **Use Case Name** | **Log Out.** |
| Primary Actor | Administrator. |
| Brief Description | In this use case user administrator kills its login session and get out from the system. If he or she want to go back in system them he or she will re-login to use the system, once the session is kill user must re-login to use the system. |
| Preconditions | Administrator is already login. |
| Basic Flows | Administrator simply clicks on the logout link to kill or destroy its login session. |
| Alternative Flows | System will redirect to login page. |

**3.3.2 Dynamic Model: Sequence Diagrams**

The dynamic model describes the behavior of a distributed parameter system in terms of how one qualitative state can turn into another. A qualitative state is described by a static model, i.e. the distributions and intersections of the qualitative fields at a particular time instant or interval. The sequence Diagram is best way to show Dynamic Model here is one diagram which shows the working of School Management System.

**Sequence Diagram**

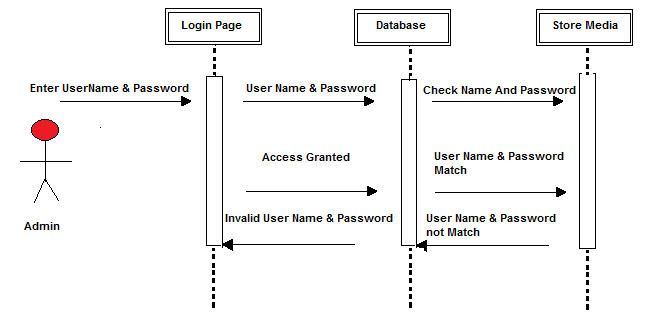


Fig 3.6 Sequence Diagram for Login Page

**Delete Record**

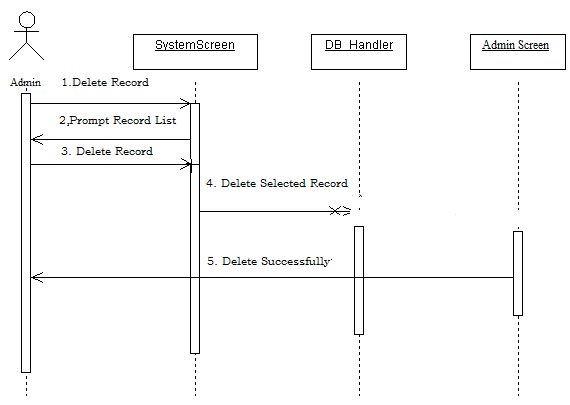
****

Fig 3.7 Sequence Diagram for Delete Record

**Insert Record**

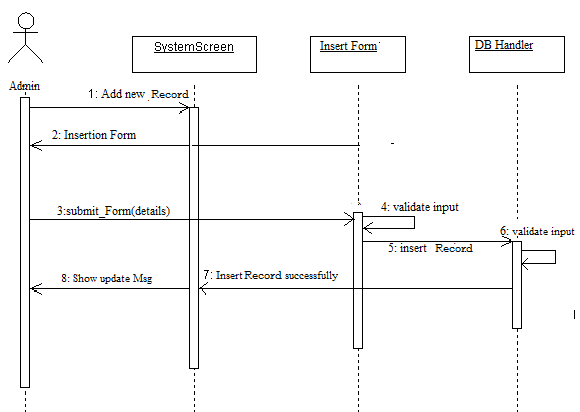
****

Fig 3.7 Sequence Diagram for Insert Record

* 1. **Database Design**

Database design is the process of producing a detail data model of database. The logical data model contains all the needed logical and physical design choice and physical storage parameters needed to generate a design in a data definition language. This can be used to create a database.

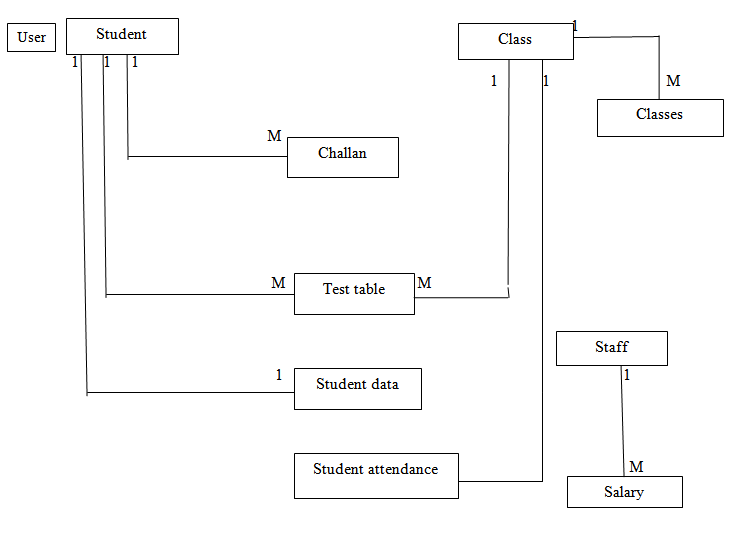
The term database design can be describe many different parts of the design of an overall database system. Principally and most correctly it can be thought of as the logical design of the database structure used to store the data. In the relational model these are the tables and views. However the term database design could also be used to apply to the overall data base application with the database management system DBMS.

There are several steps involved in developing an effective design. Regarding what data element must be stored who will access them and how.

The next step is to define the logical database. This phase does not deal with how the data will be stored physically but with how information is grouped logically. The requirements are translated into a model that provides a level of abstraction from the physical database representing data in terms of business entities and relationship rather than in terms of tables and columns.

Physical design is the final phase in which individual data elements are given attributes and defined as columns in tables. This phase also deals with performance considerations relating to the creation of indexes rollback segments temporary segments and physical layout of data file on disk. DDL (Data Definition Language) scripts are written to create database objects and to be capacity planning.

**3.4.1 Entity Relationship Diagrams (ERDs)**

****

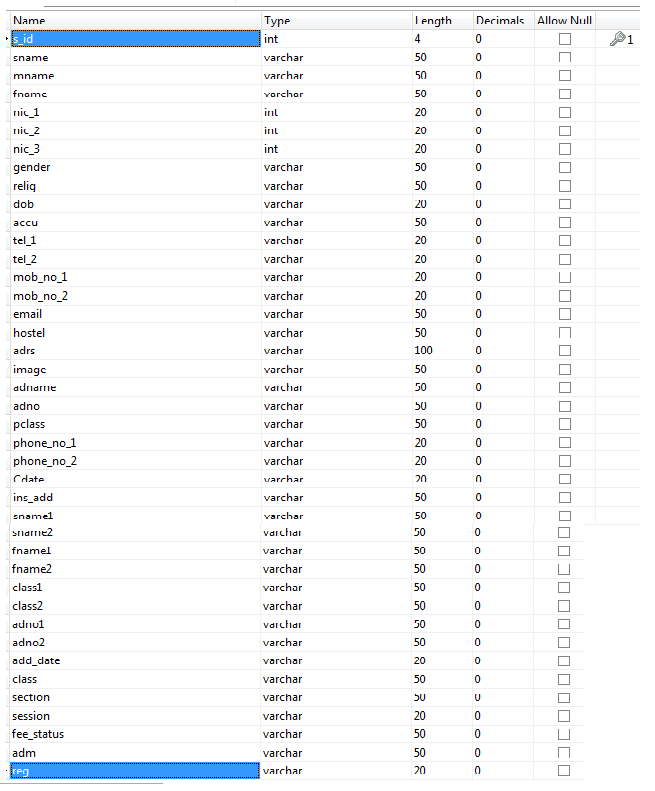
**Database Table**

**Table Name** Student

**Primary Key** student\_id

**Purpose:** This table stores the information about student at time of registration.

**Candidate Registration**

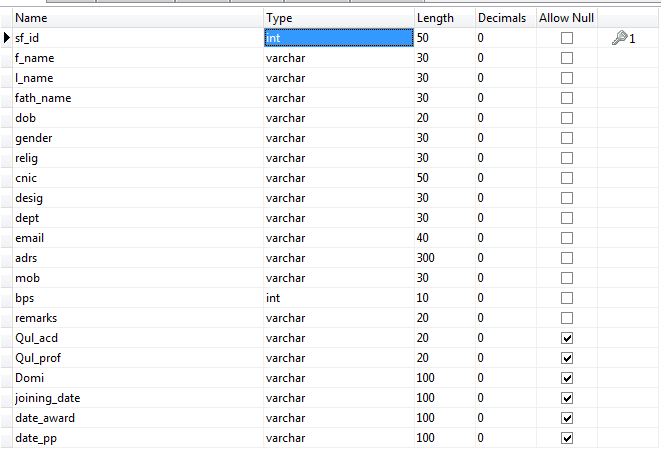
****

**Table Name** Staff

**Primary Key** sf\_id

**Purpose** This table stores the information about Staff as the time of joining.

**Staff Record**

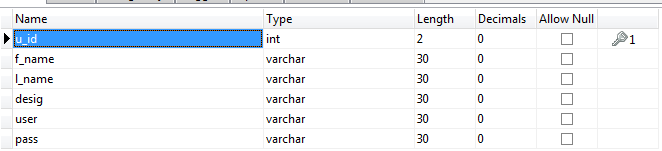
****

**Table Name** User

**Primary Key** u\_id

**Purpose** This table stores the information about system User account.

**Users**

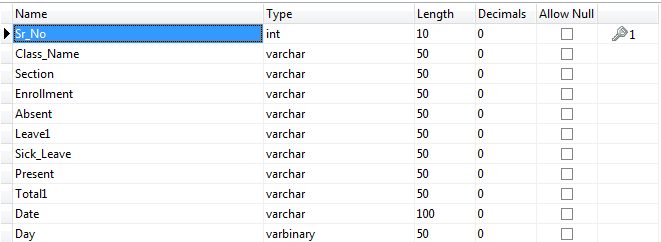
****

**Table Name**Stu**\_**attendance

**Primary Key** Sr\_No

**Purpose** This table store the information about daily student attendance.

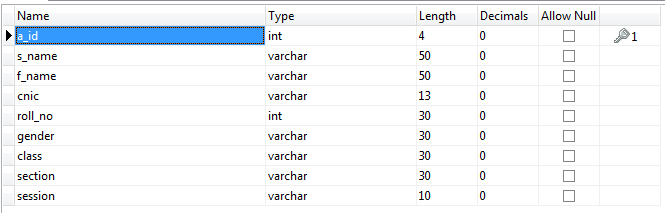
**Student Attendance**



**Table Name** stud\_data

**Primary Key** a\_id

**Student Data**



**Table Name** testable

**Primary Key** testId

**Purpose** This table stores the information about student result. **Test Table**

