Symbiosis Institute of Technology

A DBMS Project Report on

Integrated Student Management System

CAMPUS NEXUS

Submitted by

Abhishek Rajput (22070122007)

Aditya Raj (22070122009)

Archit Patil (22070122025)

Arnav Jain (22070122030)

Under the Guidance of

Prof. Vijayshree Khedkar

Department of Computer Science

SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

Index

1. Introduction

In today's expanding landscape it is crucial to have a streamlined and comprehensive approach to managing administrative and academic functions. Our project introduces a Database Management System (DBMS) specifically designed to meet these requirements. Our system offers a solution for student management incorporating key functionalities such as fee management, hostel accommodation, student records, course administration, library management, timetable scheduling,faculty coordination, attendance tracking, branch management and grade management.

The fee management module of our system simplifies transactions while ensuring transparency and efficiency. Managing hostel accommodations becomes effortless with our process for allocating and maintaining student housing. The student module acts as a repository for all information related to students – an essential component for smooth administration. Course administration is enhanced by features that facilitate planning and management.

A pivotal feature of our system is the timetable scheduling function that optimizes resource utilization and minimizes scheduling conflicts – resulting in benefits for both students and faculty members. The faculty coordination module provides a platform, for managing faculty information and schedules – an aspect of smooth academic operations.

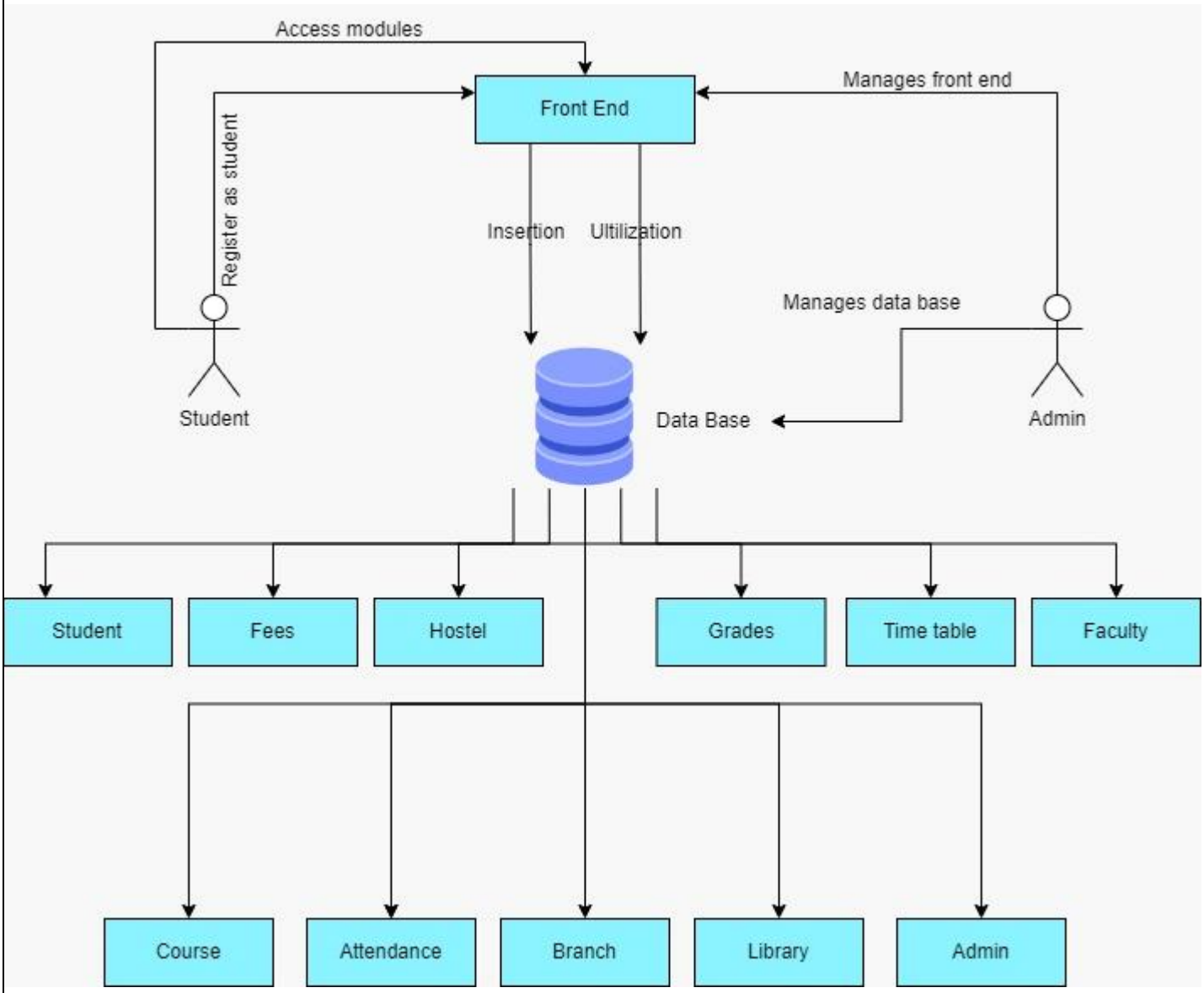
Attendance tracking is automated to ensure accuracy and convenience while the grading system guarantees an efficient process, for evaluating student performance.

This comprehensive student management system, driven by a database management system (DBMS) goes beyond being a technological solution. It is a tool, which simplifies processes while significantly improving the quality of education delivery.

2. Problem Statement

Educational institutions face a challenge in managing their academic operations due to their complex and diverse needs. The current methods used which are often fragmented and manual result in inefficiencies and errors. These issues have an impact on both management and students. Some critical problems include the handling of student data, inefficient management of courses and faculty well as the absence of a unified system for crucial processes such as fee payment, attendance tracking and grade assessments. These inefficiencies not affect effectiveness but also compromise the quality of education delivered. Moreover, the absence of real time data access and analytics severely limits decision making and strategic planning. Therefore, there is a need for an integrated student management system that can consolidate functionalities into one platform. This will significantly improve efficiency while enhancing the experience, for both students and staff members.

3. System Architecture and Modules



Modules

1. **Login Module:**
   - Authenticates users based on their username, password and grants access to either the admin or student module based on their role.

2. **Admin Module:**
   - Manages all the modules and add and remove the data of other modules.

3. **Student Module:**
   - Display comprehensive information about students, including their name, PRN (Personal Roll Number), phone number, date of birth, academic year, batch, address, etc.

4. **Fees Module:**
   - Manage financial aspects of students, categorizing them by branches, tracking payment status (paid/unpaid), and handling any fines incurred.

5. **Hostel Module:**
   - Organize information related to hostel accommodation, differentiating between boy's and girl's hostels, and categorizing accommodations as simple or luxury.

6. **Timetable Module:**
   - Create and manage class schedules with details such as timing, subjects, assigned faculty, and room numbers for effective organization and communication.

7. **Attendance Module:**
   - Monitor and record attendance data, capturing details like student PRN, percentage, and the date of attendance.

8. **Faculty Module:**
    - Maintain information about faculty members, including the branch in which they teach and relevant details, contributing to efficient faculty management.

9. **Grades Module:**
    - Track and manage student grades by associating them with GPA (Grade Point Average) for mid and end semester.

10. **Course Module:**
    - Maintain a catalog of courses with essential details such as course ID, course name, duration, and the branch ID of the course.

11. **Library Module:**
    - Organize library-related data, including categories of books, details of book issue, and return transactions.

12. **Branch Module:**
    - Implement functionalities to manage branches such as adding, updating, or deleting branch records in the Branch table.

4. Functional Requirements

1. **Login:**
    - Provide a clear interface with options for "Admin Login" and "Student Login".
    - Admin login: Upon selection, direct the user to the admin login page.
    - Student Login: Upon selection, direct the user to the student login page.

2. **Admin:**
    - Should be able to implement data management capabilities for efficient handling of student, course, library, branch, timetable, fees, hostel, grades, attendance, and faculty modules.
    - Data Addition: Admin should be able to add data in any module.
    - Data Modification: The system should support the modification of other modules.
    - Data Deletion: The system should allow the deletion of module data.
    - Data Retrieval: Admin should be able to view the complete details of modules.

3. **Student:**
    - Add Student: The system should allow the addition of new student records with details like name, address, contact information, etc.
    - Update Student: The system should allow updating information of existing student.
    - Delete Student: The system should support the removal of student records.
    - View Student Details: Users should be able to view the complete details of a student.

4. **Courses:**
    - Add Course: The system should allow the addition of new courses with details like course ID, name, and duration.
    - Update Course: The system should support the modification of existing course details.
    - Delete Course: The system should support the removal of courses.
    - View Course Details: Users should be able to view the complete details of a course.

5. **Library:**
    - Book Management: The system should manage the inventory of books including adding, updating, searching and deleting book records.
    - Issue/Return : Students and faculty should be able to issue/return books.

6. **Timetable:**
    - Edit Timetable: The system should edit class timetables for students and faculty based on course schedules.
    - View Timetable: Users should be able to view their respective timetables.

7. **Fees:**
    - Fee Payment: The system should allow students to view the payment of fees.
    - View Fine Status: Users should be able to view their fine on fee payment.

8. **Hostel:**
    - Room Allocation: The system should manage the allocation of hostel rooms to students.
    - View Hostel Details: Users should be able to view information about the hostel.

9. **Grades:**

- Enter Grades: The system should be able to enter grades for students based on their performance in assignments, exams, etc.
- View Grades: Students should be able to view their grades for different exam.

10. **Attendance:**

- View Attendance Report: Students should be able to view their attendance records.

11. **Faculty:**

- Add Faculty: The system should allow the addition of new faculty members.
- Update Faculty: The system should be able to update existing faculty information.
- Delete Faculty: The system should support the removal of faculty records.
- View Faculty Details: Users should be able to view the complete details of a faculty member.

12. **Branch:**

- Add Branch: Allow administrators to add new branches to the system.
- Update Branch: Provide functionality to update existing branch information such as branch name.
- Delete Branch: Allow administrators to remove branches from the system.
- View Branch Details: Students should be able to view complete details of a branch, including branch name and any associated information.

5. Entities, Relationships and Attributes

Entities	Relationship	Entities
Admin	Admin can Manage Attendance.	Attendance
Admin	Admin can Manage Grades.	Grades
Admin	Admin can Manage Course.	Course
Admin	Admin can Manage Fees.	Fees
Admin	Admin can Manage Faculty.	Faculty
Admin	Admin can Manage Library.	Library
Admin	Admin can Manage Hostel.	Hostel
Admin	Admin can Manage Timetable.	Timetable
Student	Student is Taught by a faculty.	Faculty
Student	Student has grade records.	Grades
Student	Student has attendance.	Attendance
Student	Student can Enroll in courses.	Courses
Student	Student Pays Fees.	Fees
Student	Student Visits Library.	Library
Student	Student Lives in Hostel.	Hostel
Student	Student Follows Timetable.	Timetable
Admin	Admin can access Admin login.	Admin Login
Student	Student can access Student login.	Student Login

Attributes:

1. **Admin:**
    - AdminID
    - Name
        - First Name
        - Last Name
    - Email

2. **Student**
    - PRN
    - Name
        - First Name
        - Last Name
    - Phone Number
    - DOB
    - Semester
    - Year
    - Branch
    - Email
    - Father’s Name
    - Address
        - City
        - Zip Code
        - State
        - Street
            - Street Name
            - Street Number

3. **Admin Login:**
    - AdminID
    - AdminPassword

4. **Student Login:**
    - PRN
    - StudentPassword

5. **Hostel**
    - PRN
    - Gender
    - Luxury Room
    - Simple Room

6. **Course**
    - CourseID
    - Course Name
    - Duration
    - Branch ID

7. **Library**
    - LibraryID
    - Category
    - Issue
    - Return

8. **Timetable**
    - TimetableID
    - Course ID
    - Faculty ID
    - Room Number
    - Timing

9. **Faculty**
    - Faculty ID
    - Branch ID
    - Faculty Name

10. **Attendance**
    - PRN
    - Percentage
    - Date

11. **Grades**
    - PRN
    - Midsem
    - Endsem

12. Fees

- PRN
- Paid/Unpaid
- Fine

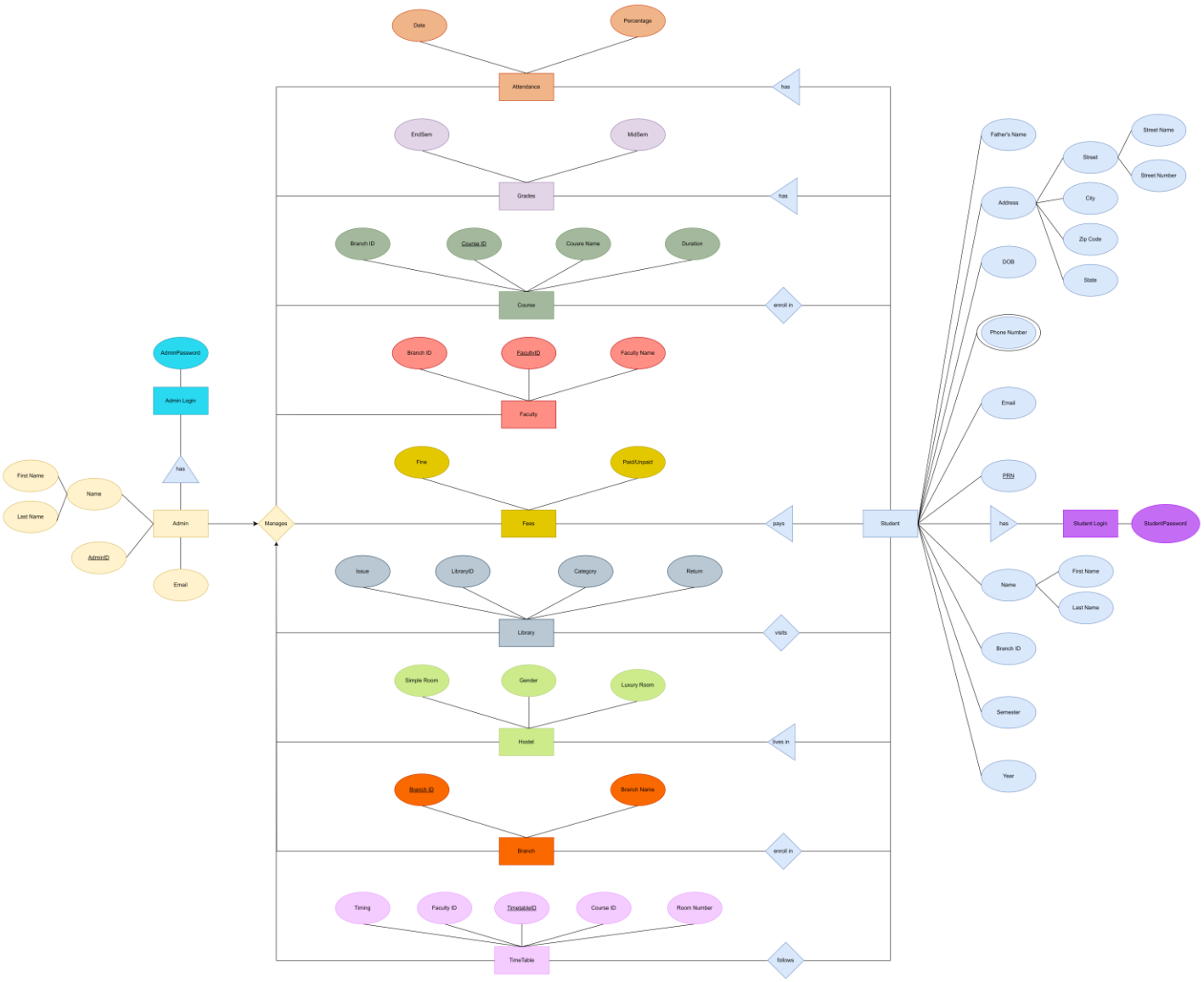
13. Branch

- Branch ID
- Branch Name

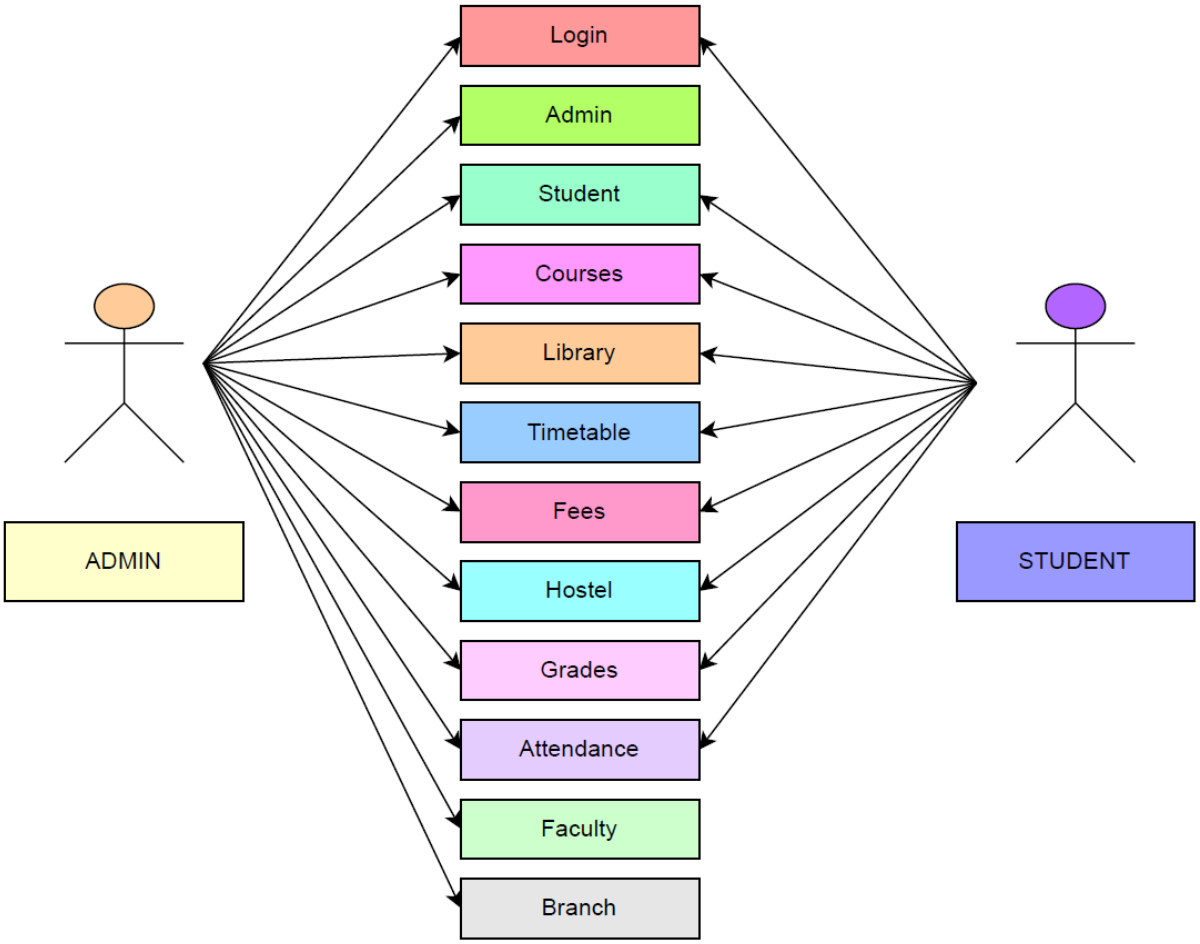
14. Phone number

- PRN
- Phone Number 1
- Phone Number 2

6. E-R Diagram


Date
Percentage
Attendance
has
EndSem
MidSem
Grades
has
Branch ID
Course ID
Cousre Name
Duration
Course
enroll in
Branch ID
FacultyID
Faculty Name
Faculty
Fine
Paid/Unpaid
Fees
pays
Issue
LibraryID
Category
Return
Library
visits
Simple Room
Gender
Luxury Room
Hostel
lives in
Branch ID
Branch Name
Branch
enroll in
Timing
Faculty ID
TimetableID
Course ID
Room Number
TimeTable
follows
AdminPassword
Admin Login
has
First Name
Name
Last Name
AdminID
Admin
Email
Manages
Student
Father's Name
Address
Street Name
Street
Street Number
City
Zip Code
State
DOB
Phone Number
Email
PRN
has
Student Login
StudentPassword
Name
First Name
Last Name
Branch ID
Semester
Year

7. Use Case Diagram



8. Relational Schema

Underlined + Bold – Primary Key
Italics – Foreign Key

1. Admin

Admin ID	Name	FirstName	LastName	Email

- **Primary Key:** Admin ID
- **Foreign Keys:** Null
- **Candidate Keys:** PRN, (Email)
- **Alternate Keys:** Email

2. Student

PRN	Name	FirstName	LastName
Year	*Branch ID*	Semester	DOB
Father's name	Email	Phone Number	Address-Street-Street Name
Address-Street-Street Number	Address-Zip code	Address-State	Address-City

- **Primary Key:** PRN
- **Foreign Keys:** Branch ID
- **Candidate Keys:** PRN, (Phone Number, Email, Address)
- **Alternate Keys:** Phone Number, Email, Address

3. Phone Number

PRN	Phone Number 1	Phone Number 2

- **Primary Key:** PRN
- **Foreign Keys:** PRN
- **Candidate Keys:** PRN, (Phone Number)
- **Alternate Keys:** Phone Number

4. AdminLogin

Admin ID	AdminPassword

- **Primary Key:** Admin ID
- **Foreign Keys:** Admin ID
- **Candidate Keys:** PRN, (AdminPassword)
- **Alternate Keys:** AdminPassword

5. StudentLogin

PRN	StudentPassword

- **Primary Key:** PRN
- **Foreign Keys:** PRN
- **Candidate Keys:** PRN, (StudentPassword)
- **Alternate Keys:** StudentPassword

6. Hostel

PRN	Gender	Simple Room	Luxury Room

- **Primary Key:** PRN
- **Foreign Keys:** PRN
- **Candidate Keys:** PRN
- **Alternate Keys:** Null

7. Course

Course ID	Course Name	*Branch ID*	Duration

- **Primary Key:** Course ID
- **Foreign Keys:** Branch ID
- **Candidate Keys:** Course ID, (Course Name)
- **Alternate Keys:** Course Name

8. Library

Library ID	Category	Issue	Return

- **Primary Key:** Library ID
- **Foreign Keys:** Null
- **Candidate Keys:** Library ID
- **Alternate Keys:** Null

9. Time Table

Timetable ID	*Course ID*	*Faculty ID*	Room Number	Timing

- **Primary Key:** Timetable ID
- **Foreign Keys:** Course ID, Faculty ID
- **Candidate Keys:** Timetable ID, (Faculty ID)
- **Alternate Keys:** Null

10.Faculty

Faculty ID	*Branch ID*	Faculty Name

- **Primary Key:** Faculty ID
- **Foreign Keys:** Branch ID
- **Candidate Keys:** Faculty ID
- **Alternate Keys:** Null

11.Attendence

PRN	Date	Percentage

- **Primary Key:** PRN
- **Foreign Keys:** PRN
- **Candidate Keys:** PRN
- **Alternate Keys:** Null

12.Grades

PRN	Mid Sem	End Sem

- **Primary Key:** PRN
- **Foreign Keys:** PRN
- **Candidate Keys:** PRN
- **Alternate Keys:** Null

13. Fees

PRN	Paid/Unpaid	Fine

- **Primary Key:** PRN
- **Foreign Keys:** PRN
- **Candidate Keys:** PRN
- **Alternate Keys:** Null

14. Branch

Branch ID	Branch Name

- **Primary Key:** Branch ID
- **Foreign Keys:** Null
- **Candidate Keys:** Branch ID, (Branch Name)
- **Alternate Keys:** Branch Name

9. Application of Codd's Rules

The rules satisfied by our relational model are:

1. **Information Rule:** Data stored in Relational model must be a value of some cell of a table.
   i. This rule is followed in our database as Our SMS organizes all data, such, as fee details, student information and hostel accommodations in an efficient manner using tables.

2. **Guaranteed Access Rule**: Every data element must be accessible by table name, its primary key and name of attribute whose value is to be determined.
   i. This rule is followed in our database as In our system each piece of data like a students course details or accommodation information can be accurately retrieved through its table name, key and attribute name.

3. **Systematic Treatment of Null Values**: NULL values in the database only correspond to missing, unknown or not applicable values.
   i. This rule is followed in our database Our SMS consistently represents missing, unknown or inapplicable information across all datasets using values.

4. **Active Online Catalog**: Structure of database must be stored in an online catalogwhich can be queried by authorized users.
   i. This rule is followed in our database as The database schema of our SMS is stored in an catalog. This catalog can be queried through SQL to fulfill the requirements of this rule.

5. **Comprehensive Data Sub Language Rule**: A database can only be accessed using a language having a linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.
   i. This rule is followed in our database as Our system utilizes SQL as a language that supports all aspects of data manipulation, definition and transaction management for interacting with the database.

6. **View Updating Rule**: Different views created for various purposes should beautomatically updatable by the system.
   i. This rule is not followed in our database. Although updating the view willupdate the table used for creating it, it is not recommended by most of the database. Hence this rule is not used in most of the databases.

7. **High Level Insert, Update and Delete Rule:** Relational Model should support insert,delete, update etc. operations at each level of relations. Also, set operations like Union, Intersection and minus should be supported.
   i. Our Integrated SMS follows this rule by allowing set based operations, for data manipulation. For example it supports bulk updating of student records. Inserting entries simultaneously.

8. **Physical Data Independence:** Any modification in the physical location of a tableshould not enforce modification at application level.
   i. This rule is followed in our database as Our system ensures that changes in the storage of data do not affect the user interface or application performance. We prioritize maintaining this independence.

9. **Logical Data Independence:** Any modification in logical or conceptual schema of atable should not enforce modification at application level. For example, merging of two tables into one should not affect application accessing it which is difficult to achieve.
   i. This rule is not followed in our database. But in an ideal scenario, this is difficult to achieve since all the logical and user views will be tied so stronglythat they will be almost the same.

10. **Integrity Independence:** A database must be independent of the application that usesit. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.
   i. This rule is not followed in our database. Because suppose we change the constraint to put a minimum value or change a range of any data then the application must also then show messages or errors accordingly. Although,primary key and foreign key constraints are maintained in our database.

11. **Distribution Independence:** Distribution of data over various locations is not visibleto end-users.
   i. This rule is not followed in our database. Since we are using the MySQL community edition, this condition is violated. For a distributed database, wewill have to use MySQL cluster.

12. **Non-Subversion Rule:** If a system has an interface that provides access to low-levelrecords, then the interface must not be able to subvert the system and bypass securityand integrity constraints.
   i. Our SMS adheres to this rule by ensuring that accessing low level records via SQL does not bypass established security and integrity constraints, within the system.

In summary our Student Management System showcases a commitment, to upholding the principles of Codd's database rules. This solid foundation ensures its strength and efficiency, as an educational management tool.