PROJECT PROPOSAL

ON

Student Management System

Guided By:
Mr. Anuj kumar

Created By:
Yashvi Chaudhary(AF04991261)
Charu(AF04991721)
Vanshika(AF04992208)

Table Of Contents

Index

PROJECT TITLE:

Student Management System (Terminal-Based)

Using Java, JDBC, MySQL

INTRODUCTION:

Student information plays a key role in the smooth functioning of any educational institute. Handling student records manually involves maintaining admission details, personal information, academic data, and contact details in registers or spreadsheets. As the number of students increases, managing these records becomes slow, error-prone, and difficult to update.

A Student Management System helps overcome these issues by computerizing the entire record-keeping process. This project focuses on building a terminal-based application using Java, JDBC, and MySQL that stores and manages student data in an organized way. The system makes it easier to add, update, delete, and search student information while ensuring accuracy, quick access, and better security of data.

OBJECTIVE:

- To automate student-related operations, including storing and managing personal, academic, and contact details.
- To reduce paperwork and avoid manual mistakes in maintaining student records.
- To store all student information in a secure and well-organized database.
- To ensure fast searching and retrieval of student details whenever needed.
- To arrange student data in a structured way for easier access and management.
- To systematically track updates such as new admissions, modifications, and deletions of student records.
- This aims to provide a simple interface so administrators can easily operate the system.

PROJECT CATEGORY:

- This project belongs to the Database Management System category.
- It is a terminal based software application.
- Core Java is used as the primary programming language.
- Java Database Connectivity (JDBC) is used to connect database with java.
- MySQL is used as the relational database management system.
- The system demonstrate CRUD operations (i.e. create, retrieve, update and delete operations).

ANALYSIS:

STUDENT MANAGEMENT SYSTEM: MODULES

1. Admin Management
2. Student Management
3. Course Management
4. Marks Management
5. Report & Search Management

Module 1: Admin Management

1.1– Admin Login
1.2 – Admin Profile Update
1.3 – View System Overview / Dashboard

Module 2: Student Management

2.1 – Student Registration
2.2 – Student Update
2.3 – Student List
2.4 – Student Details / Profile
2.5 – Student Delete

Module 3: Course Management

3.1 – Add Course
3.2 – Update Course
3.3 – Course List
3.4 – Course Delete

Module 4: Marks Management

4.1 – Add Marks
4.2 – Update Marks
4.3 – Marks List
4.4 – View Student Marks
4.5 – Delete Marks

Module 5: Report & Search Management

5.1 – Search Student by ID
5.2 – Search Student by Name
5.3 – Generate Student Report
5.4 – Course-wise Report
5.5 – Performance Summary

DATABASE DESIGN:

Table 1: Admin

Fields	datatype	properties
admin_id	int	Primary Key, auto_increment
username	varchar(50)	Not null , unique
password	varchar(255)	Not null (store hashed password)
name	varchar(100)	Not null
created_at	timestamp	default current_timestanp

Table 2: Student

Fields	datatype	properties
student_id	int	Primary Key, auto_increment
first_name	varchar (100)	Not null
last_name	varchar(100)	Null
email	varchar(150)	Not null , unique
dob	date	null
gender	enum('m','f','o')	default 'o'
phone	varchar(20)	null
address	varchar(300)	null
created_at	timestamp	Default current_timestamp

Table 3: Course

Fields	datatype	properties
course_id	int	Primary Key, auto_increment
course_name	varchar(150)	Not null
course_code	varchar(50)	Not null , unique
duration	varchar(50)	null (e.g., "1 year", "6 months")
fee	decimal(10,2)	null
created_at	timestamp	Default current_timestamp

Table 4: Enrollment

Fields	datatype	properties
enrollment_id	int	Primary Key, auto_increment
student_id	Int	Not null, foreign key→ Student(student_id)
course_id	int	Not null, foreign key Course(course_id)

Fields	datatype	properties
enroll_date	date	Not null
status	enum('active','completed','dropped')	default 'active'
created_at	timestamp	Default current_timestamp

Table 5: Marks

Fields	datatype	properties
marks_id	int	Primary Key, auto_increment
student_id	int	not null, foreign key → Student(student_id)
course_id	int	not null, foreign key→ course(course_id)
subject	varchar(150)	Not null
marks_obtained	int	null
total_marks	int	null
grade	varchar(5)	null
exam_date	date	null
created_at	timestamp	Default current_timestamp

Table 6: Report_log

Fields	datatype	properties
log_id	int	Primary Key, auto_increment
admin_id	int	Null, foreign key → Admin(admin_id)
action	varchar(200)	Not null (e.g., "Added Student", "Updated Marks")
target_id	int	null (id of student/course/record affected)
timestamp	timestamp	Default current_timestamp
remarks	varchar(300)	null

Relationships:

- **Student** 1 — * N **Enrollment** — * N **Course** (many-to-many via Enrollment)
- **Student** 1 — * N **Marks** (a student can have many marks records)
- **Course** 1 — * N **Marks**
- **Admin** 1 — * N **Report_Log** (admin actions audit)

Table 7: Admin↔Report_log

(One → Many)

Relationship	Description
Admin Report_Log	One admin can perform many actions that get recorded in the log.
Foreign Key	Report_Log.admin_id → Admin.admin_id

Table 8: Student ⟷ Enrollment

(One → Many)

Relationship	Description
Student → Enrollment	A student can enroll in many courses.
Foreign Key	Enrollment.student_id → Student.student_id

Table 9: Course ⟷ Enrollment

(One → Many)

Relationship	Description
Course → Enrollment	A course can have many enrolled students.
Foreign Key	Enrollment.course_id → Course.course_id

Table 10: Student ⟷ Marks

(One → Many)

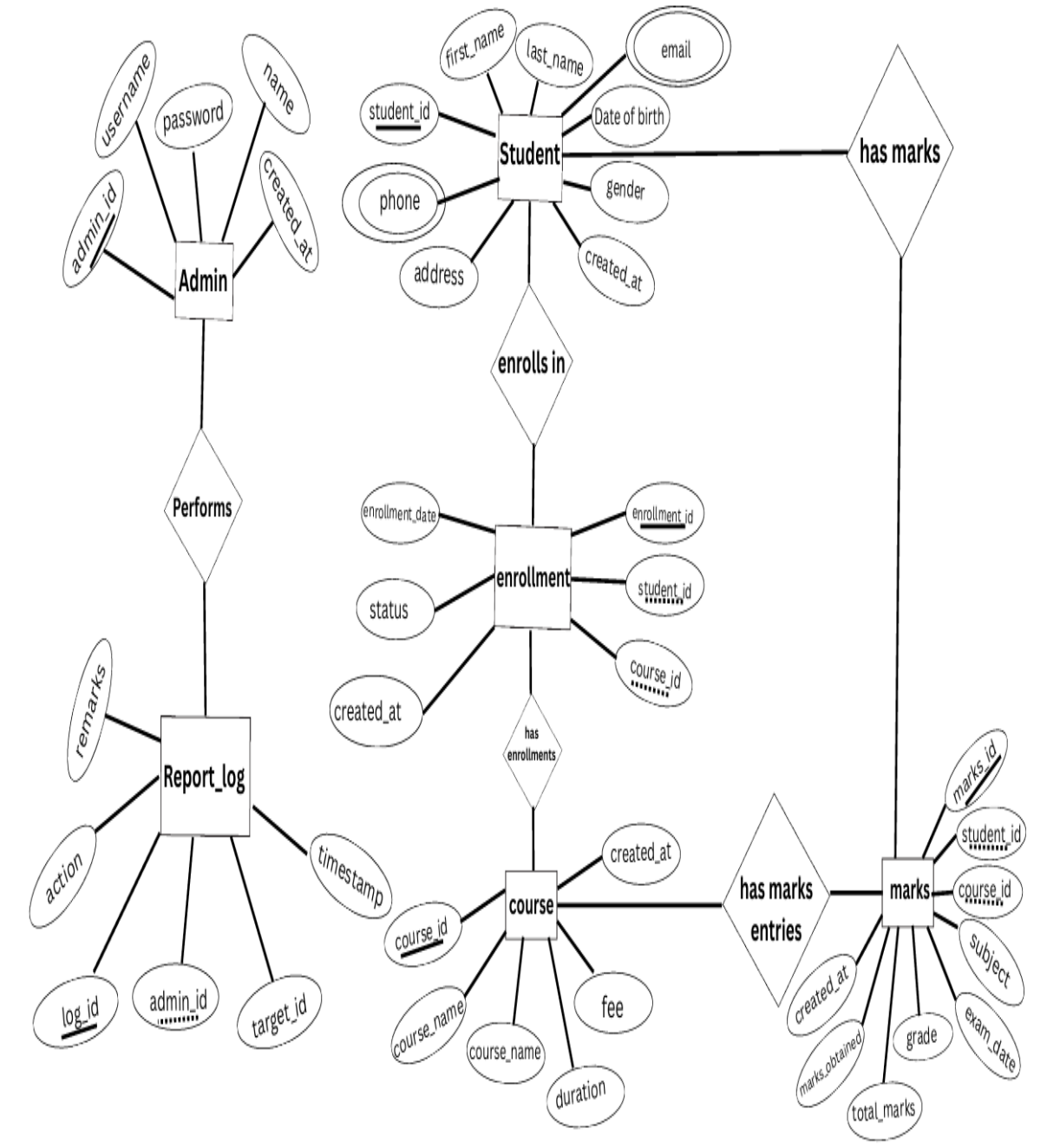
Relationship	Description
Student → Marks	A student can have multiple marks records for different subjects and exams.
Foreign Key	Marks.student_id → Student.student_id

Table 11: Course → Marks

(One → Many)

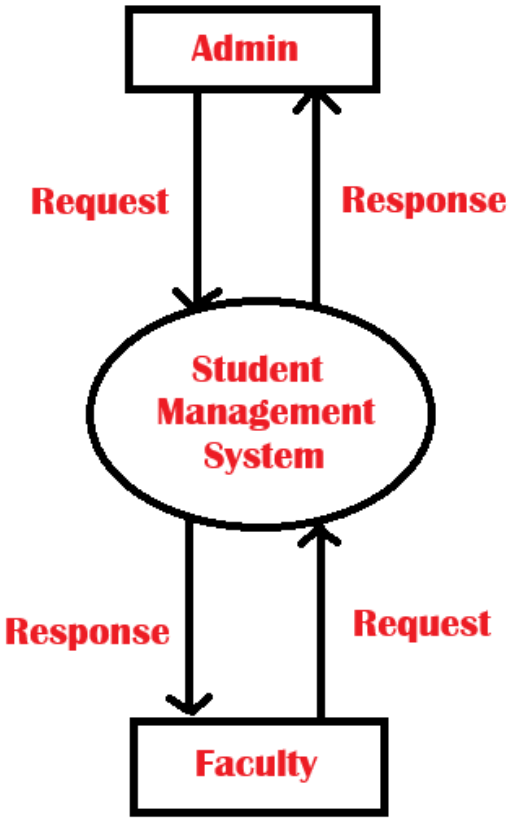
Relationship	Description
Course→ Marks	A course can have multiple marks entries for students enrolled in it.
Foreign Key	Marks.course_id → Course.course_id

ENTITY RELATIONSHIP DIAGRAM:

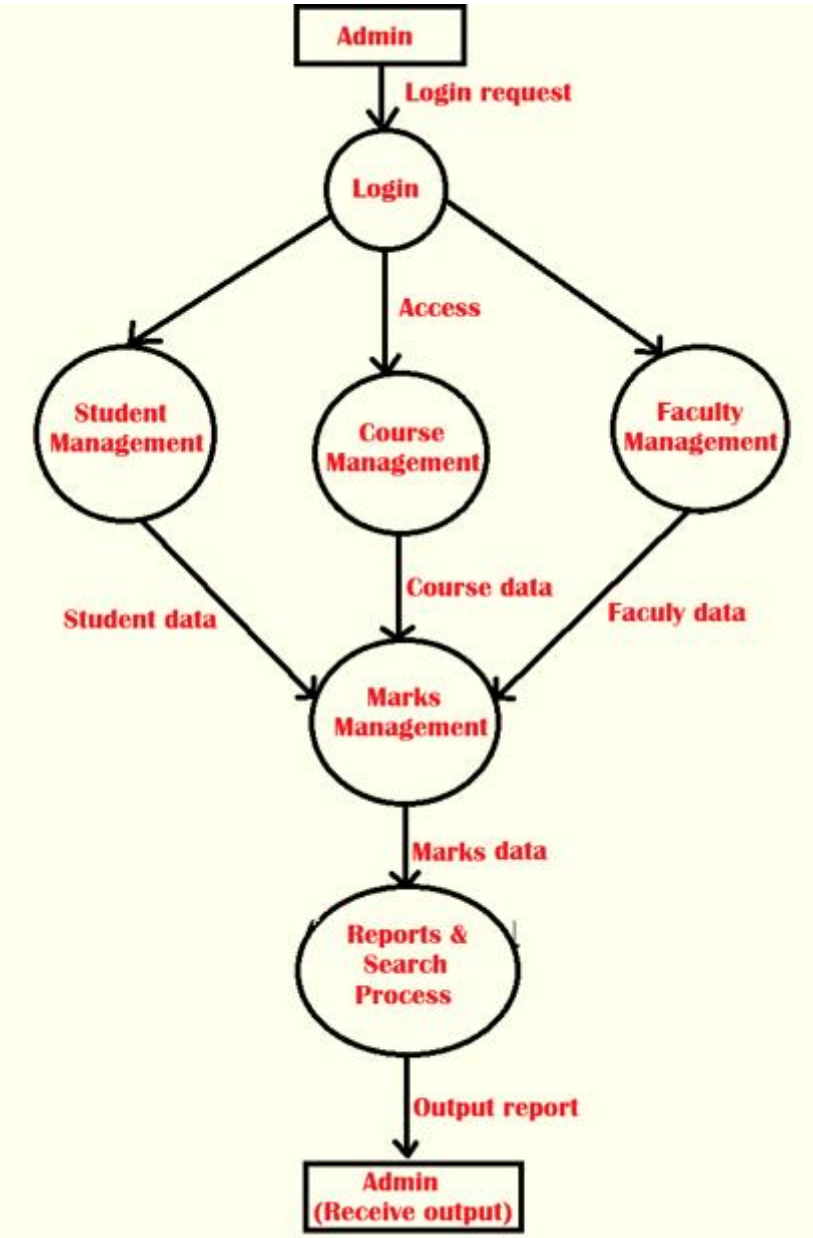


DATA FLOW DIAGRAM:

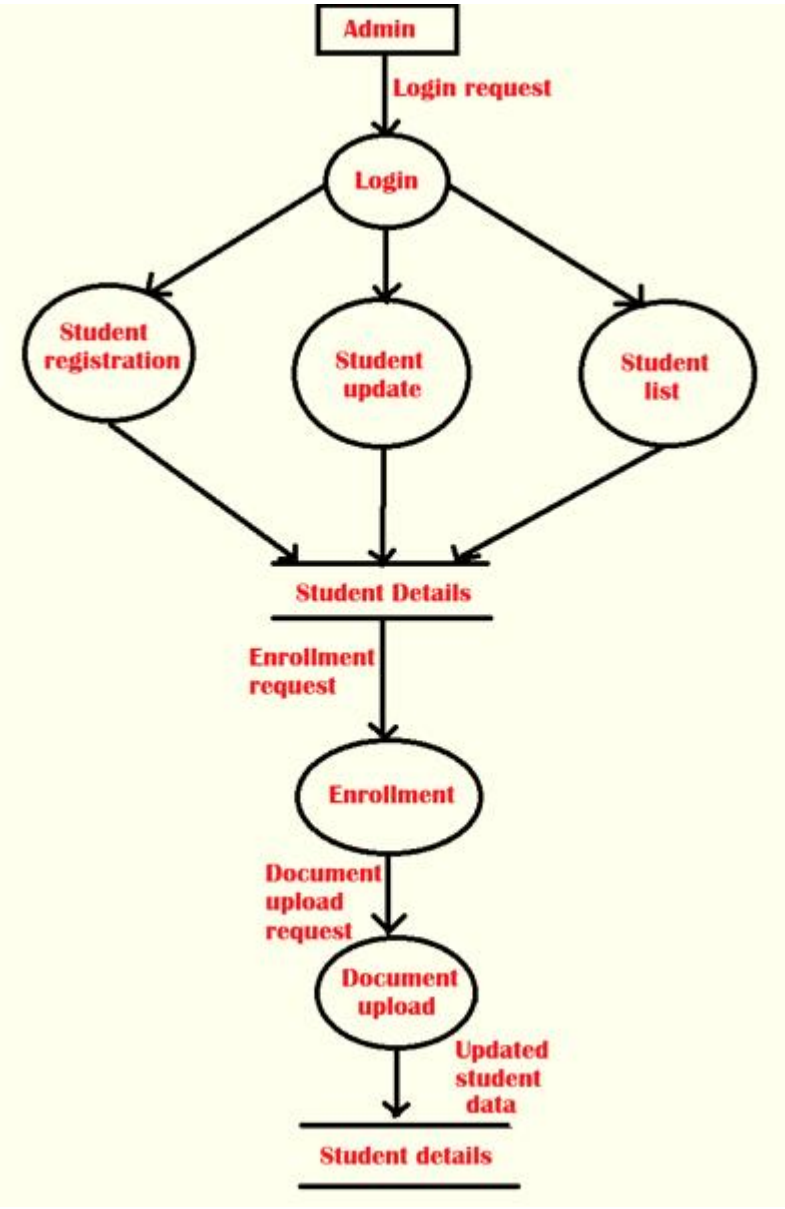
Zero level DFD



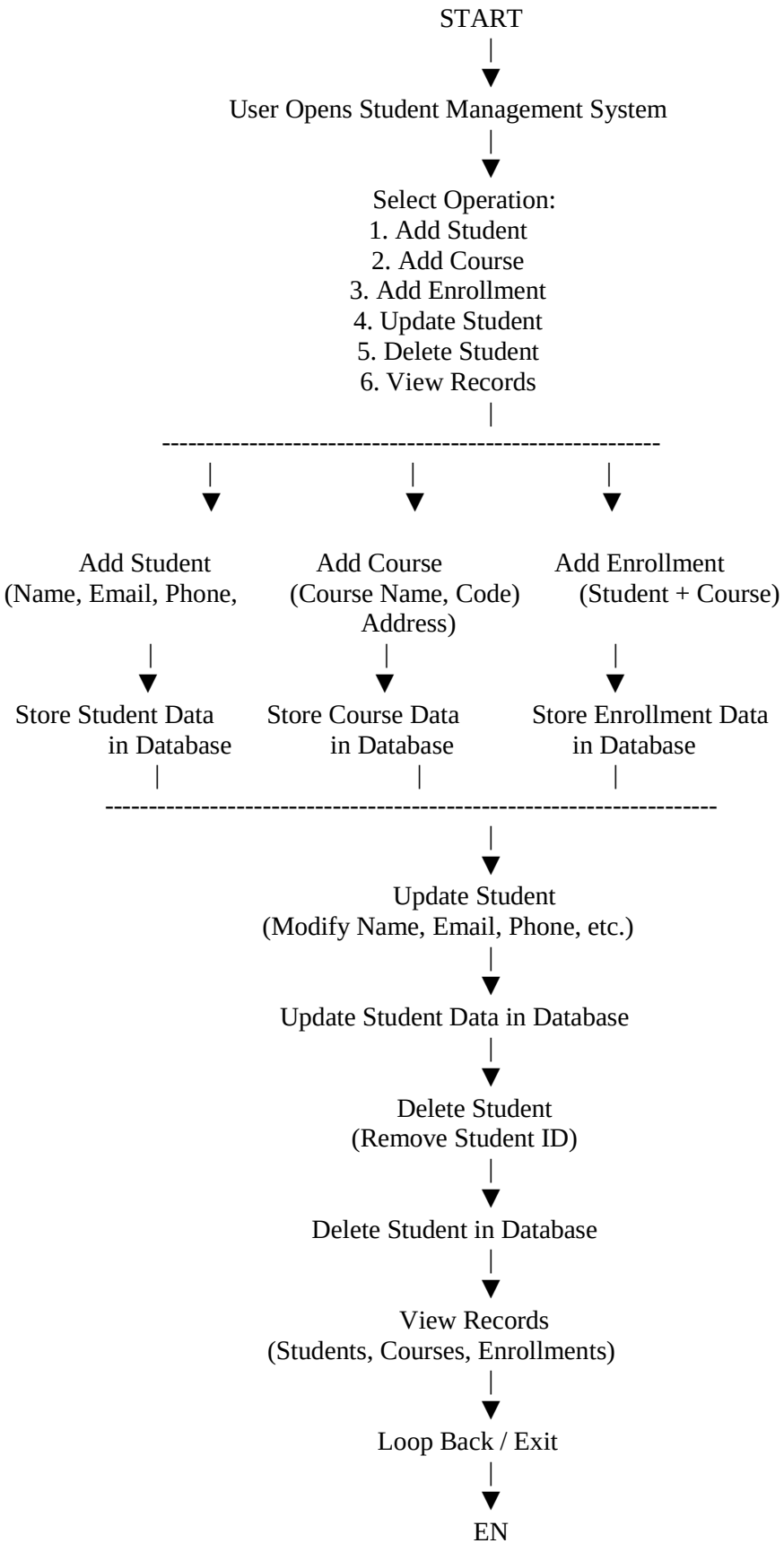
First level DFD



Second level DFD



PROCESS LOGICAL DIAGRAM:



PLATFORM USED:

a) Hardware Requirements

- Processor: Intel i3 or higher
- RAM: Minimum 4GB
- Hard Disk: 500MB free space
- Operating System: Windows/Linux/Mac

b) Software Requirements

- JDK 8 or above
- MySQL Server
- IDE: Eclipse / IntelliJ IDEA / NetBeans
- JDBC Driver
- OS: Windows 10/11

FUTURE SCOPE:

- **Integration with Web or Mobile Application:**
  The system can be upgraded to a web-based or mobile application to allow access from anywhere.
- **Role-Based Access System:**
  Admin, teachers, and students can be given separate login access with different permissions.
- **Automated Attendance System:**
  Future versions can include biometric or RFID-based attendance tracking for accuracy.
- **Performance Analytics & Reports:**
  The system can generate automatic performance charts, grade analysis, and student progress reports.
- **Notification and Alerts:**
  SMS or Email alerts for attendance shortage, exam schedules, and fee reminders can be added.
- **Cloud Storage Integration:**
  Student data can be stored securely on cloud platforms for accessibility and safety.
- **AI-Based Recommendation System:**
  AI can be used to suggest courses or activities based on student performance and interest.
- **Online Fee Management:**
  Integration with payment gateways to allow online fee submission and receipt generation.
- **Homework & Assignment Module:**
  Teachers can upload assignments and students can submit them through the system.
- **Parent Portal:**
  A separate parent login can be developed to check attendance, performance, and notices.

BIBLIOGRAPHY:

- Oracle Java Documentation
- MySQL Official Documentation
- JDBC API Guide
- Reference books on DBMS and Java Programming
- Online educational resources (GeeksforGeeks, TutorialsPoint, JavaTPoint)