Student Database Management System Final Project Portfolio

Name: Usaidkhan Pathan Nasimkhan

Banner ID: 916489191

Project URL: http://elvis.rowan.edu/~usaidk28/AdvancedDatabases/Final_Project/

1. Table of Contents

2. Requirements Collection & Analysis

The project began with identifying the real-world need for an integrated **Student Database Management System** in educational institutions. Through functional decomposition and actor-role modelling, I identified critical users such as **students**, **instructors**, **admins**, and **finance officers**, and outlined use cases for each.

Key Activities:

- **Gathered over 20 functional/non-functional requirements**, including course registration, fee tracking, grade management, and attendance.
- Conducted research on similar platforms like Banner and PeopleSoft to understand industry standards.
- Created detailed **use-case scenarios** covering each actor's interactions with the system.
- Defined **business rules and data constraints**, such as course limits per semester and valid fee payment statuses.

System Goals:

- Provide a secure, web-accessible platform for managing student data.
- Automate processes such as fee payment status updates and grade logging.
- Maintain data integrity and reduce redundancy through normalization.

3. Conceptual & Logical Design

This phase focused on designing the system's data model, starting from an Entity-Relationship Diagram (ERD) and transforming it into a normalized logical schema. The process ensured that the structure supported all identified requirements without redundancy or anomalies.

ERD Highlights:

- Defined **8+ primary entities**: Student, Course, Instructor, Enrollment, Fee, Grade, Attendance, Scholarship.
- Established correct **cardinalities** (1:N, M:N) and used **foreign keys** for referential integrity.
- Developed relationship tables like **ENROLLMENT**, **COURSE_INSTRUCTOR**, and **STUDENT_SCHOLARSHIP** to handle M:N relationships.

Normalization Process:

- **1NF:** Removed repeating fields like CourseEnrolled in Student.
- **2NF:** Moved semester/year to separate STUDENT_COURSE to avoid partial dependencies.
- **3NF:** Separated Scholarship from Fee and Student to remove transitive dependencies.

Deliverables:

- Final **Relational Schema** with 11+ normalized tables.
- Justified normalization choices with real-world examples and dependency analysis.

4. Physical Design & Performance

In this phase, the conceptual schema was implemented physically using **MySQL Workbench**. Tables were created with all defined fields, data types, constraints, and indexing for performance optimization.

Implementation Details:

- Used **Forward Engineering** to generate SQL scripts from the EER diagram.
- Created schema usaid and all required tables with **foreign key constraints**, **default values**, and **data types**.
- **Inserted test data** with at least 5 records per table (Student, Instructor, Course, Enrollment, Attendance, Fee, Grade, etc.).
- Ensured **referential integrity** by inserting parent table records before dependent records.

Advanced Features Implemented:

- **Views** for reporting: StudentCourseSummary, FeeStatusOverview.

- **Stored Procedures**: EnrollStudentInCourse (with rollback) and UpdateFeeStatus.
- **Functions**: GetTotalFeeForStudent, GetAverageGradeForStudent.
- **Triggers**: Auto-update FeeStatus after payment and assign Incomplete grade after enrollment.

Performance Notes:

- Used **indexing** on commonly joined fields like StudentID, CourseID to improve query speed.
- Ensured **atomicity** using transactions in procedures to prevent inconsistent states.

5. Frontend Design & Implementation

The final component involved connecting the backend database to a user-friendly frontend built using **PHP, HTML, and CSS**, hosted on the **Rowan Elvis web server**.

Features:

- **Responsive user interface** with navigation bar, footer, and clean sectioning.
- CRUD pages: Add/Edit/Delete/View Students, Courses, Fees, Grades, etc.
- Secure **role-based access control** for different user types.
- Embedded **stored procedure/function calls** to provide real-time updates (e.g., showing total fee).

Functional Pages:

- **Student Dashboard**: Displays personal information, grades, attendance, fee history.
- **Admin Panel**: Allows full control over user and academic data.
- **Instructor Dashboard**: Add/edit grades and attendance records.
- **Fee Payment Page**: Lists due payments, tracks scholarships, and updates via triggers.

Hosting & Deployment:

- **Deployed live** at: http://elvis.rowan.edu/~usaidk28/AdvancedDatabases/Final_Project/
- Managed permissions via chmod 755 and structured files into organized directories.
- Integrated **PHP + MySQL** seamlessly with mysqli_connect and error handling.