University Database System

Ubaid Bin Waris	2212416
Mushaid Hussain	2212408
Muhammad Salman	2212400
Jehanzeb Khalid	2212391

Table of contents

Introduction

The University Database System is designed to efficiently manage essential academic data within a structured relational database. It organizes crucial information, including courses, classrooms, faculty, and schedules, all while enhancing data accessibility and maintaining data integrity. By reducing redundancy, the system ensures consistency across various university functions and prevents errors in data entry. With robust CRUD operations, administrators can easily create, retrieve, update, and delete records as needed. The database also supports complex queries, allowing for comprehensive reporting on classroom usage, faculty schedules, and course assignments.

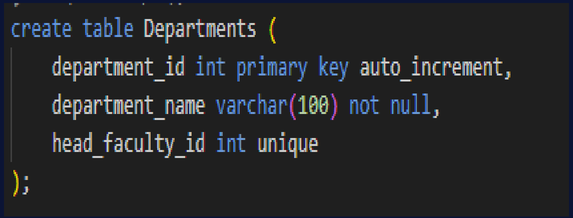
Implementation of project

- Table Structure: Each table in the schema represents a key entity (e.g., Students, Faculty, Courses, Departments) and defines the attributes relevant to that entity.
- Primary Keys: Each table includes a primary key to uniquely identify each record (e.g., StudentID for Students, CourseID for Courses), ensuring data integrity.
- Foreign Keys: Relationships between tables are established through foreign keys (e.g., StudentID in the Enrollments table linking to the Students table), enabling data referencing across entities.
- Relationships: The schema outlines one-to-many and many-to-many relationships (e.g., one department can have many courses, and students can enroll in multiple courses) to accurately reflect real-world connections.
- Constraints and Integrity: The schema enforces data constraints (e.g., NOT NULL, UNIQUE) to maintain referential integrity and ensure valid data entries throughout the system.

Database Schema

Departments Table

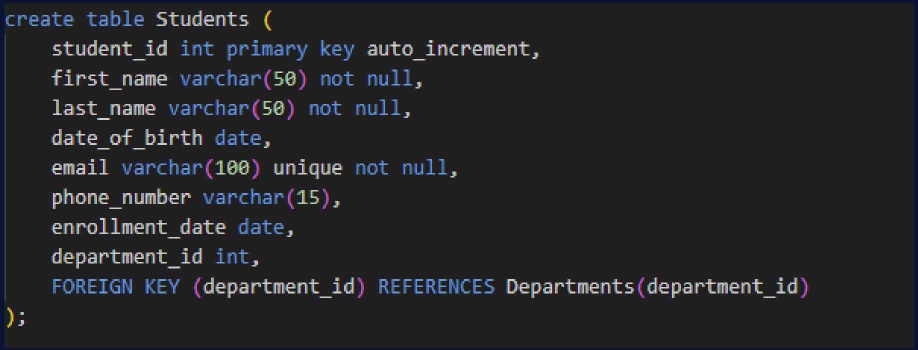
Stores information about departments within the university, including a unique department ID and the head of the department.



```sql
create table Departments (
    department_id int primary key auto_increment,
    department_name varchar(100) not null,
    head_faculty_id int unique
);
```

Student Table

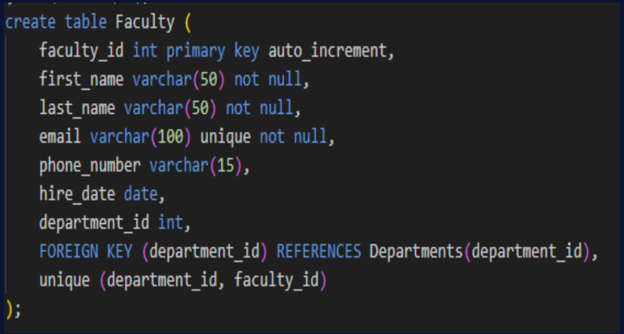
Contains information about students, including personal details and the department they're enrolled in.



```sql
create table Students (
    student_id int primary key auto_increment,
    first_name varchar(50) not null,
    last_name varchar(50) not null,
    date_of_birth date,
    email varchar(100) unique not null,
    phone_number varchar(15),
    enrollment_date date,
    department_id int,
    FOREIGN KEY (department_id) REFERENCES Departments(department_id)
);
```

Faculty Table

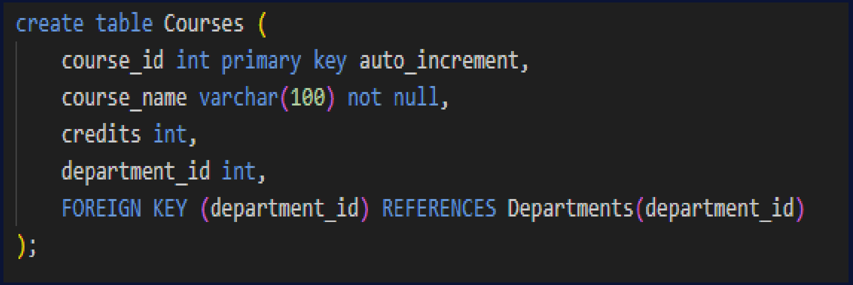
Contains information about faculty members, including personal details and the department they are associated with.



```
create table Faculty (
    faculty_id int primary key auto_increment,
    first_name varchar(50) not null,
    last_name varchar(50) not null,
    email varchar(100) unique not null,
    phone_number varchar(15),
    hire_date date,
    department_id int,
    FOREIGN KEY (department_id) REFERENCES Departments(department_id),
    unique (department_id, faculty_id)
);
```

Course Table

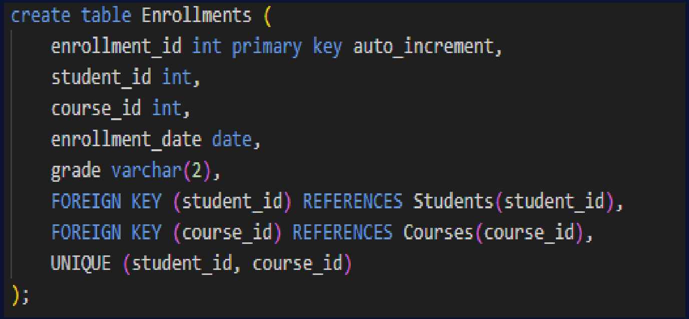
Stores details about courses, including the department offering the course.



```
create table Courses (
    course_id int primary key auto_increment,
    course_name varchar(100) not null,
    credits int,
    department_id int,
    FOREIGN KEY (department_id) REFERENCES Departments(department_id)
);
```

Enrollment Table

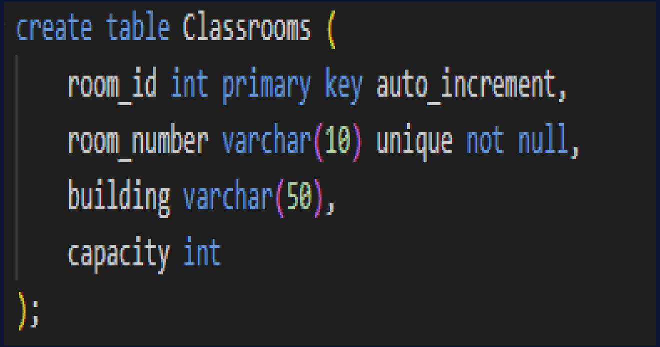
Tracks course enrollments for students, including grades. This is a weak entity dependent on both Students and Courses.



```
create table Enrollments (
    enrollment_id int primary key auto_increment,
    student_id int,
    course_id int,
    enrollment_date date,
    grade varchar(2),
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id),
    UNIQUE (student_id, course_id)
);
```

Classroom Table

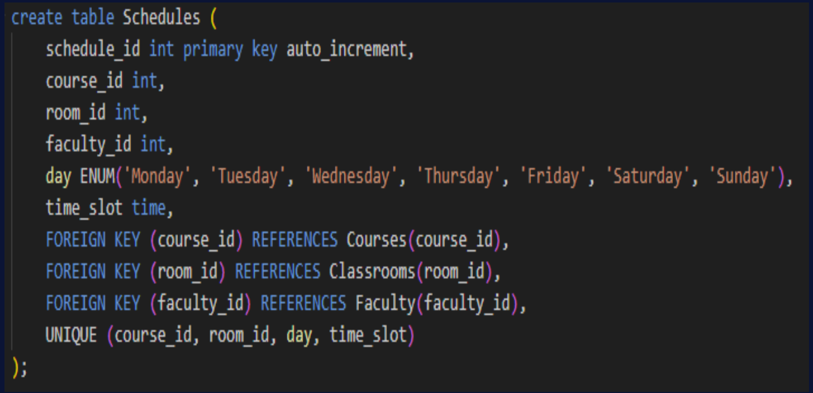
Manages classrooms, including unique room numbers and attributes like building and capacity.



```
create table Classrooms (
    room_id int primary key auto_increment,
    room_number varchar(10) unique not null,
    building varchar(50),
    capacity int
);
```

Schedules Table

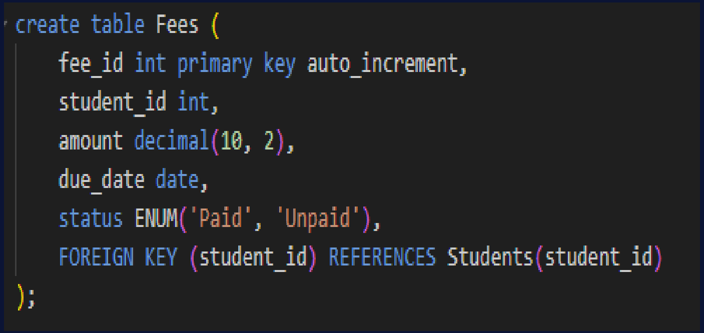
Schedules classes, linking them to courses, classrooms, and faculty members. This is a weak entity dependent on both Courses and Faculty.



```
create table Schedules (
    schedule_id int primary key auto_increment,
    course_id int,
    room_id int,
    faculty_id int,
    day ENUM('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'),
    time_slot time,
    FOREIGN KEY (course_id) REFERENCES Courses(course_id),
    FOREIGN KEY (room_id) REFERENCES Classrooms(room_id),
    FOREIGN KEY (faculty_id) REFERENCES Faculty(faculty_id),
    UNIQUE (course_id, room_id, day, time_slot)
);
```

Fees Table

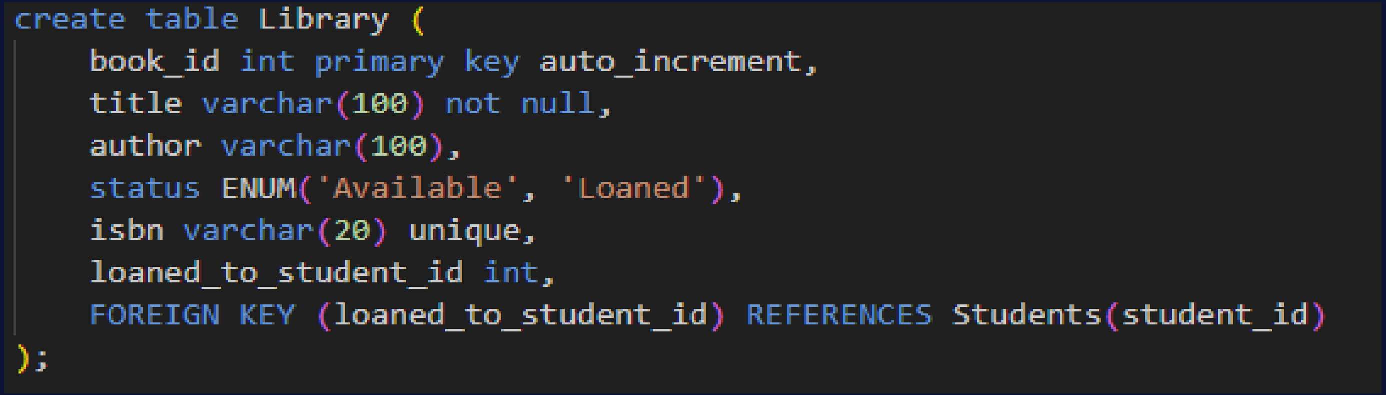
Tracks fees for each student, with status indicating whether fees are paid or unpaid.



```
create table Fees (
    fee_id int primary key auto_increment,
    student_id int,
    amount decimal(10, 2),
    due_date date,
    status ENUM('Paid', 'Unpaid'),
    FOREIGN KEY (student_id) REFERENCES Students(student_id)
);
```

Library Table

Manages books in the university’s library, including loaned books and their status.

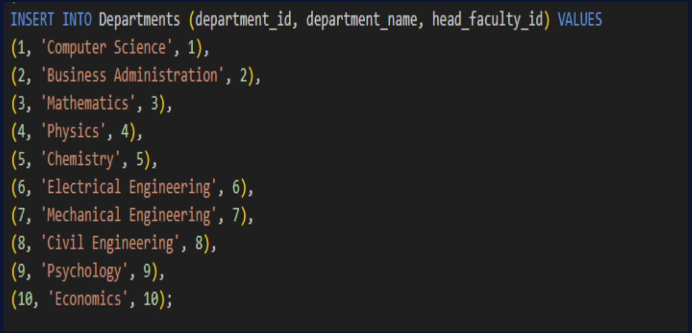


```sql
create table Library (
    book_id int primary key auto_increment,
    title varchar(100) not null,
    author varchar(100),
    status ENUM('Available', 'Loaned'),
    isbn varchar(20) unique,
    loaned_to_student_id int,
    FOREIGN KEY (loaned_to_student_id) REFERENCES Students(student_id)
);
```

Data Manipulating CRUD Operations

Insert Data

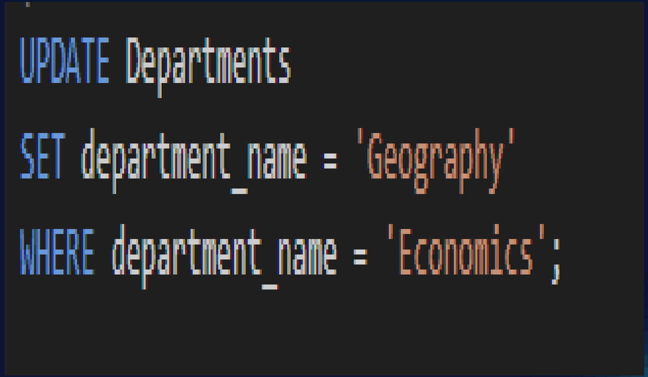
To populate tables with initial data, INSERT INTO commands are used. Each table requires specific values based on its attributes and foreign key dependencies.



```
INSERT INTO Departments (department_id, department_name, head_faculty_id) VALUES
(1, 'Computer Science', 1),
(2, 'Business Administration', 2),
(3, 'Mathematics', 3),
(4, 'Physics', 4),
(5, 'Chemistry', 5),
(6, 'Electrical Engineering', 6),
(7, 'Mechanical Engineering', 7),
(8, 'Civil Engineering', 8),
(9, 'Psychology', 9),
(10, 'Economics', 10);
```

Update Data

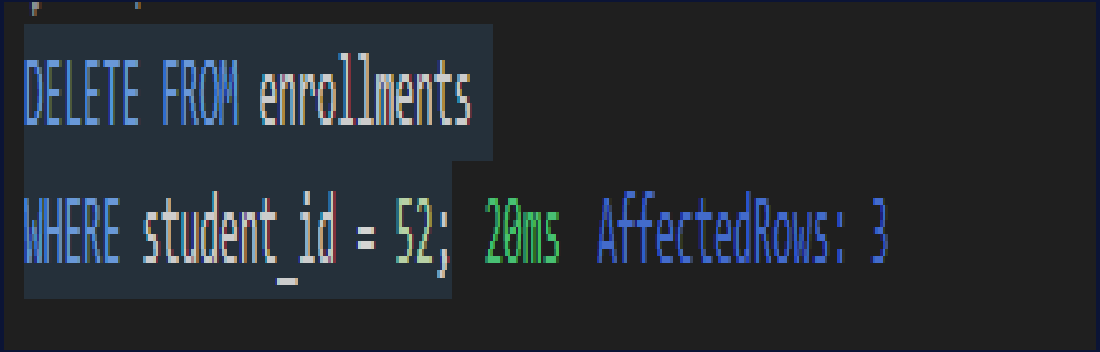
The UPDATE command allows modifying existing data in the tables, commonly used for updating contact details or course assignments.



```
UPDATE Departments
SET department_name = 'Geography'
WHERE department_name = 'Economics';
```

Delete Data

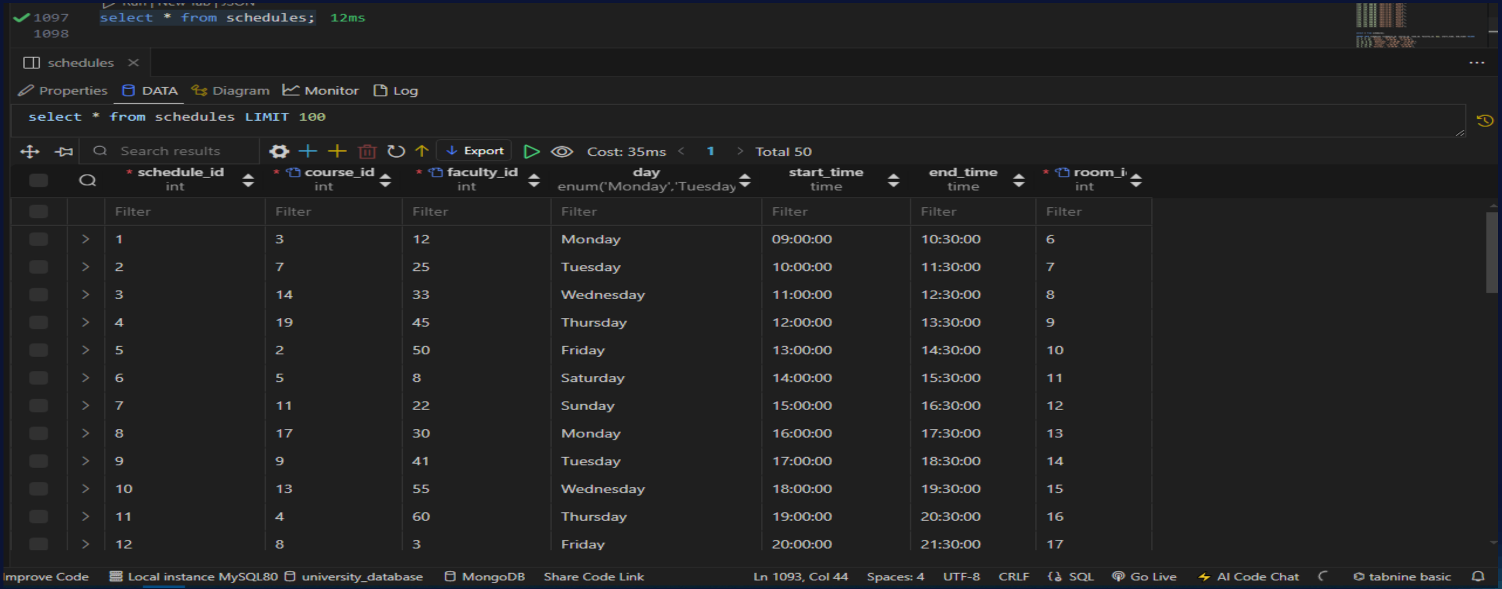
To delete data from the University Database System, you can use the SQL DELETE statement, specifying the table and conditions for the records to be removed. For example, to delete a student record:



```
DELETE FROM enrollments
WHERE student_id = 52;  20ms  AffectedRows: 3
```

Select Data

The SELECT command retrieves data from tables, often with JOINs to connect related tables. This is used for displaying information, generating reports, and validating data integrity.



```
select * from schedules;  12ms
```

```
select * from schedules LIMIT 100
```

schedule_id int	course_id int	faculty_id int	day enum('Monday','Tuesday	start_time time	end_time time	room_i int
1	3	12	Monday	09:00:00	10:30:00	6
2	7	25	Tuesday	10:00:00	11:30:00	7
3	14	33	Wednesday	11:00:00	12:30:00	8
4	19	45	Thursday	12:00:00	13:30:00	9
5	2	50	Friday	13:00:00	14:30:00	10
6	5	8	Saturday	14:00:00	15:30:00	11
7	11	22	Sunday	15:00:00	16:30:00	12
8	17	30	Monday	16:00:00	17:30:00	13
9	9	41	Tuesday	17:00:00	18:30:00	14
10	13	55	Wednesday	18:00:00	19:30:00	15
11	4	60	Thursday	19:00:00	20:30:00	16
12	8	3	Friday	20:00:00	21:30:00	17

Schema Relationships and Constraints

- **Primary Keys** Each table has a primary key (department_id, student_id, faculty_id, course_id, etc.) to uniquely identify records.

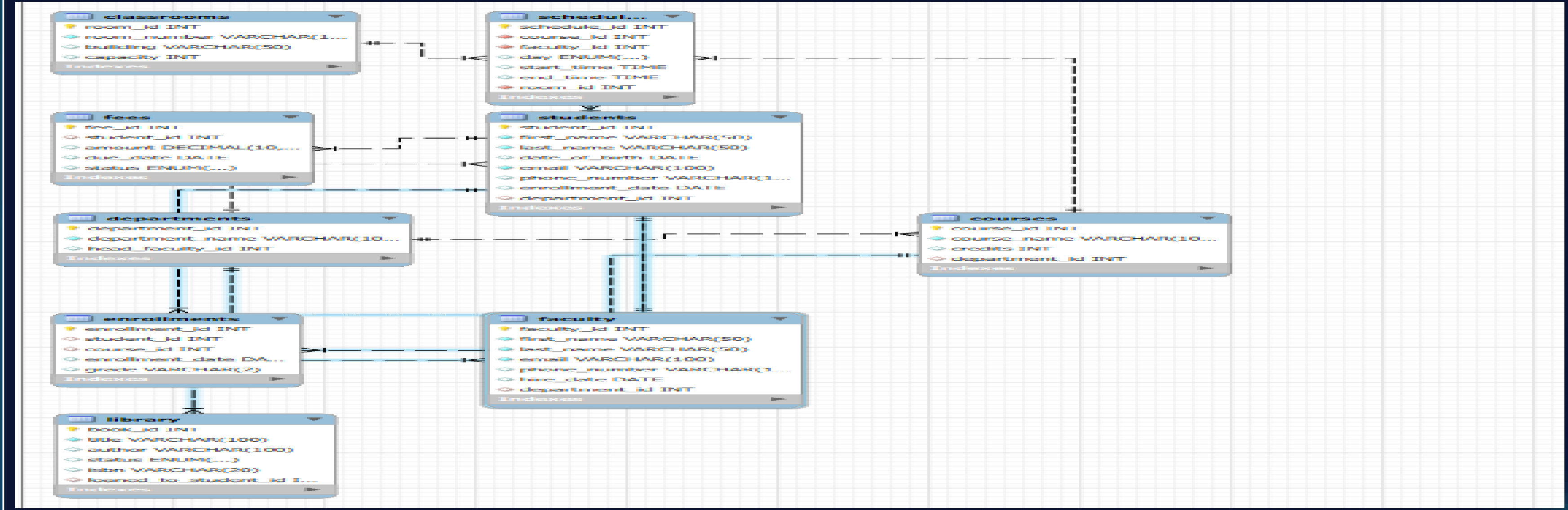
- **Foreign Keys** Foreign key constraints ensure that relationships between tables (like department_id in Faculty and Students) maintain data integrity.

- **Unique Constraints** Unique constraints, such as on email in Students and Faculty, ensure that certain attributes remain distinct across records.

- **Enumerated Data Types**: For attributes with limited values (e.g., status in Library and Fees tables), ENUM data types improve data accuracy and constraints.

ER DIAGRAM

ER Diagram



Conclusion

In conclusion, the University Database System serves as a crucial tool for effectively managing and organizing essential university data. Its well-structured schema, robust SQL functionality, and emphasis on data integrity ensure that university operations are conducted smoothly and efficiently. By demonstrating core database principles and facilitating seamless interactions between various entities, this system not only meets current administrative needs but also lays a solid foundation for future enhancements. Ultimately, it is an invaluable asset that enhances academic management and supports the ongoing growth and development of the university.

Thanks!

Do you have any questions?