

**SCHOOL OF INFORMATION SCIENCE**

**Fundamentals of Database Systems**

Active student management system

for yanet campus(6 kilo)

Section - one

**Group members** **ID**

1. Dagim asnake UGR/2281/15
2. Abiy asfaw UGR/1633/15
3. Mihret Mesfin UGR/0338/15
4. Selamawit wale UGR/6565/15

**Submitted to**: Mr. Aminu.M

**Date :**12/02/2024

**Table of contents**

1. Introduction…………………………………………………………………….………………………….1
   1. Background………………………………………….………………………………………….1
   2. Purpose of the system…………………………….……………………………………….1
   3. Statements of the problem…………………….……………………………………....1
   4. Scope of the project………………………………………………………………………..2
   5. Objective of the project……………………………….………………………………….2
      1. General objective…………………………………………………………………3
      2. Specific objectives………………………………………………………………..3
   6. Database Development Methodology…………..…………………………………4
      1. Data Sources & Collection Methods………….………………………….4
      2. DB Analysis and Design Methods …………………………………………4
   7. Deliverables of the Project…….……………………..…………………………………4
   8. Development Tools, Platforms and Technologies…………………………….5
   9. Project Time Plan………………………………………………………..…………………..5
2. Requirement Specification…………………………….……………………………………………6
   1. Data Requirements………..……………………….……………..……………………….6
   2. Transaction Requirements……..………………..……………..………………………7
      1. Data Entry Requirements……………………………………………………….7
      2. Data Retrieval Requirements………………..……………………………….8
      3. Data Updating Requirements………………..……………………………….9
      4. Data Removal Requirements………………………………………………..10
3. Database Design…………………………………………………………….…………………………11
   1. Conceptual database design of the new system……………..……………..11
      1. Entities with their description………………………………………………11
      2. Attributes with their description…………..……………………………..12
      3. Relationships between the entities……………..……………………….13
      4. E-R diagram…………………………………………………..……………………..14
   2. Logical Database Design……………………………………….………………………..15
      1. ER-Relation Mapping…………………………………..……………………….15
      2. Validating model with Normalization……………………………………16
         1. First Normal Form (1NF)……………………………………………16
         2. Second Normal form (2NF)…………………………..……………16
         3. Third Normal Form (3NF)…………………………..………………17
      3. Relational Schema with referential Integrity after normalization.17
   3. Physical database design…………………………………………………………….18
      1. Physical design strategy…………………………………………………..18
      2. Database Deployment details……….…………………………………19
4. Implementation and testing…………………………………………….………………………21
   1. SQL script for creating the database ……………………………….…………21
   2. SQL Scripts for creating the tables, view, indexes…………….…………21
   3. Testing (with sample data- for all CRUD operations)………………….23
5. References………………………………………………………………………………………………24
6. Appendix (Forms, Reports of the Organization etc.)………………………………..24

**1.Introduction**

* 1. **Background**

As part of our ongoing efforts to streamline and improve the management process for the students of yanet medical college, we have implemented a new database system to manage our student and staff actions. In this system, we aim to efficiently manage and store student information, ensuring accurate and up-to-date records for the medical campus. it has become essential to implement a streamlined management process that minimizes errors and reduces administrative burden. Our new database system allows us to collect and store student information securely, while also facilitating easy access and retrieval of data for administrative purposes. The system is built to enhance administrative tasks associated with managing students in an educational institution. It serves as a centralized hub for storing and organizing student information, academic records, communication tools, and more. The system aims to improve efficiency, accuracy, and communication among administrators, teachers and students. The system helps the campus better manage student-related tasks and provide a more effective and engaging learning experience.

* 1. **Purpose of the system**

The purpose of the active student management system for students at this medical campus in the database is to centralize and automate the student management process, ensuring accurate and timely data management. This system aims to streamline the administrative tasks, minimize errors, and reduce administrative burden by providing a comprehensive and efficient platform for entering and managing crucial personal and academic details of students. By optimizing data management and reducing manual paperwork, the system aims to improve the experience for both students and administrative staff. Some key purposes of using our system include Centralized Student Information, Streamlined Administrative Tasks, Improved Communication etc. Overall, the purpose of an active student management system is to create a more efficient, organized, and engaging learning environment for students while supporting educators and administrators in their daily tasks.

* 1. **Statements of the problem**

In the context of our system, the current manual processes for capturing, processing, retrieving, and disseminating student information are inefficient and prone to errors. The lack of a centralized and automated system leads to challenges in maintaining accurate and up-to-date student data, resulting in delays in communication, difficulty in tracking student progress, and inefficiencies in administrative tasks.

**Capture:**

- The current methods of capturing student information, such as paper-based forms and manual data entry, are time-consuming and error-prone.

- There is a lack of standardized procedures for collecting and inputting student data, leading to inconsistencies and inaccuracies in the information stored.

**Process:**

- The processing of student data, including tasks like attendance tracking, grade management, and scheduling, is predominantly manual and labor-intensive.

- Without automated processes, there is a risk of data duplication, discrepancies, and delays in updating student records, affecting the overall efficiency of academic operations.

**Retrieve:**

- Accessing student information from disparate sources is challenging due to the lack of a centralized repository for data storage.

- Retrieving specific student details quickly and accurately for decision-making purposes is hindered by the fragmented nature of information storage and retrieval processes.

**Disseminate:**

- Communication among stakeholders, including administrators, teachers, parents, and students, is hindered by the absence of a streamlined communication platform within the educational institution.

- Important updates, event notifications, and academic progress reports are not disseminated effectively, leading to miscommunication and a lack of timely information sharing.

Overall, the current student management system's deficiencies in capturing, processing, retrieving, and disseminating student information impede the efficient operation of the campus and hinder the ability to provide personalized support for student success. A comprehensive and automated system is needed to address these challenges and enhance the management of student-related tasks in an educational setting.

* 1. **Scope of the project**

The scope of this project includes the design, development, and implementation of an active registration system for students at a medical campus. The system will utilize a centralized database to streamline and automate the management process, improve data management, enhance security measures, and provide convenient accessibility for students. The system is the “Active student management System for Students of Medical Campus”. The system will be used in any campuses to get the information from their students and then storing that data for future usage.

* 1. **Objective of the project** 
     1. **General objective**

The objective of this project is to design, develop, and implement an active student management system for students at the medical campus that utilizes a centralized database that aims to streamline and automate the management process, improve data management, enhance security measures, and provide convenient accessibility for students.

* + 1. **Specific objectives**

1. **Planning Phase**:

- Define project scope, objectives, and requirements for the active student management system.

- Identify key stakeholders, including administrators, teachers, parents, and students, to gather input on system needs and functionalities.

- Develop a project plan outlining timelines, resources, and deliverables for each phase of the DSDLC(Database system development life-cycle).

1. **Analysis Phase**:

- Conduct a thorough analysis of current manual processes for capturing, processing, retrieving, and disseminating student information.

- Identify data entities, relationships, and attributes relevant to student management.

- Define system requirements based on administrators feedback and best practices in student information management.

1. **Design Phase**:

- Design a centralized database schema that efficiently stores and organizes student information, including student profiles, grades, and communication logs.

- Create data flow diagrams and entity-relationship diagrams to illustrate data flow and relationships within the system.

-Map the entity-relationship diagram that was created.

1. **Implementation Phase**:

- Build and configure the active student management system based on the defined requirements and design specifications.

- Populate the database with sample data for testing and validation.

And lastly,

1. **Testing Phase**:

- Perform comprehensive testing of the active student management system to ensure data accuracy, system functionality, and user experience.

- Conduct usability testing with stakeholders to gather feedback on system performance and usability.

- Address any identified issues or bugs through iterative testing and refinement.

* 1. **Database Development Methodology**
     1. **Data Sources & Collection Methods**

The data for the management system will be collected from various sources, including:

1. **Administrator of the campus**: Collecting the system details such as attribute requirements and general constraints from the Administrator of the campus.
2. **Students**: by asking student themselves and the data they are asked when they register and continue in their school time.
3. **Teachers**: Obtaining data on course Scheduling information, timings, and locations from the teachers that are currently teaching there.
   * 1. **Database Analysis and Design Methods**

The analysis and design of the database will involve the following steps:

1. **Entity-Relationship (ER) modeling**: Identifying the entities, attributes, and relationships between them.
2. **Normalization**: Ensuring the database is in a normalized form to minimize redundancy and improve data integrity.
3. **Schema design**: Designing the database schema based on the ER model, including tables, relationships, and constraints.
   1. **Deliverables of the Project**

The project will deliver the following key components:

1. A fully functional and user-friendly student registration database system.

2. Documentation outlining the database structure, including tables, relationships, and fields.

3. User manuals for administrators and staff on how to input, update, and access student registration information.

4. Security protocols and measures to protect student data and privacy.

5. Integration with other school systems, such as grade reporting.

6. Regular data backups and disaster recovery plans.

7. Training materials for staff on how to use the registration system effectively.

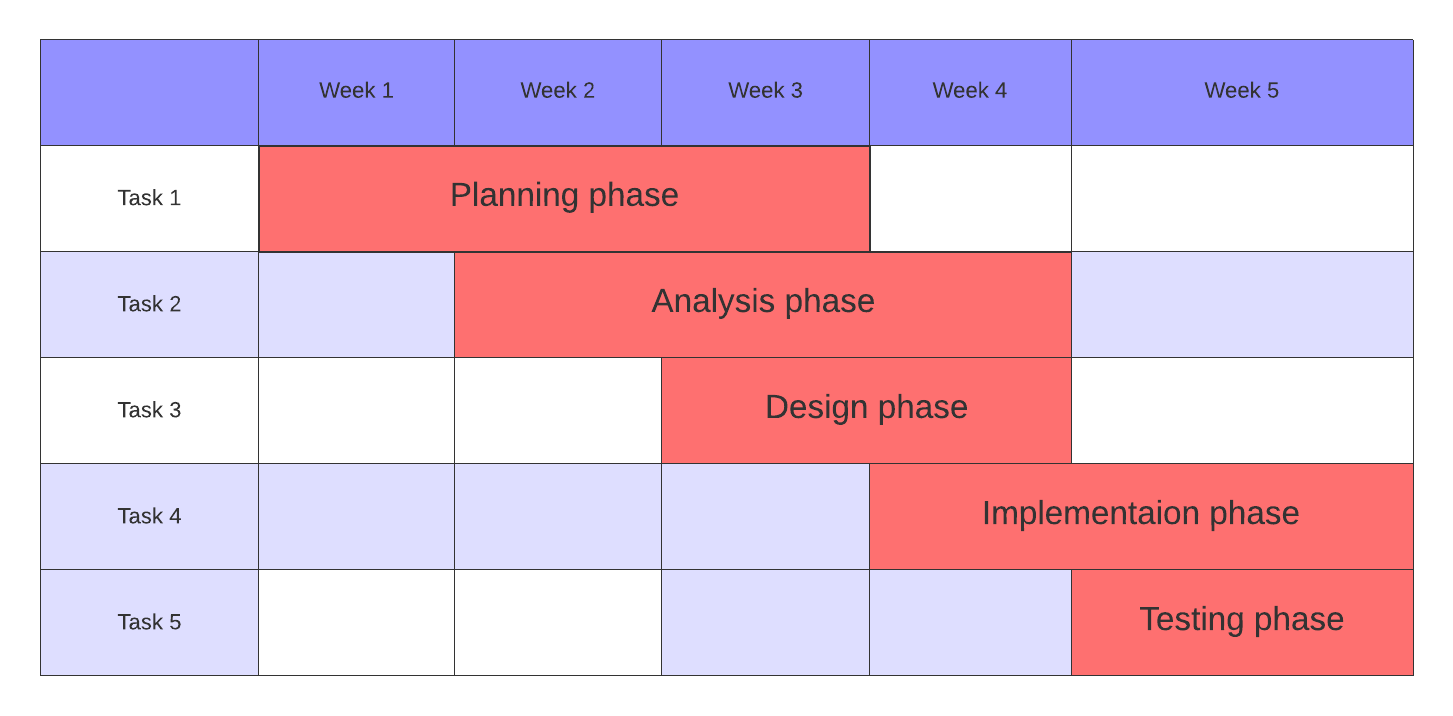
8. Ongoing technical support and maintenance for the database system.

9. Regular updates and improvements to the database system based on user feedback and changing needs.

* 1. **Development Tools, Platforms, and Technologies**

1, Relational database management system: Microsoft SQL server for database storage and management.

* 1. **Project Time Plan**
* The project time plan was developed by Gantt chart by using lucid chart.

****

1. **Requirement specification**
   1. **Data Requirements**

The data requirements of an active student management system encompass all the information necessary to effectively manage and administer student-related activities within the campus. These requirements typically include:

1. **Student Information**:

- Personal details: Name, date of birth, gender, address, contact information, etc.

- Academic details: Student ID, admission number, enrollment status, program/course enrolled in, class/section, academic year, etc.

- Emergency contact details: Next of kin, emergency contact numbers, medical information, allergies, etc.

1. **Administrative Information**:

- Enrollment and registration data: Dates of enrollment, registration status, fees paid, etc.

- Grade and assessment records: Grades, exam scores, assignments, projects, assessments, etc.

- Disciplinary records: Incidents, warnings, suspensions, expulsions, etc.

- Financial records: Fee payments, scholarships, financial aid, outstanding balances.

1. **Faculty and Staff Information**:

- Faculty details: Name, contact information, qualifications, courses taught, etc.

- Staff details: Administrative staff, support staff, roles and responsibilities, contact

information, etc.

1. **Course Information**:

- Course catalogs: Course codes, titles, descriptions, prerequisites, credit hours, etc.

- Curriculum : Course outlines, learning objectives, textbooks, reference

materials, etc.

1. **System Configuration and Settings**:

- User management: User accounts, roles, permissions, access control, etc.

- Configuration settings: System preferences, customization options, etc.

These data requirements serve as the foundation for designing and developing the that facilitates efficient student administration, academic planning, and institutional management. The system should ensure data integrity, security, and accessibility while meeting the diverse needs of students, faculty, staff, and administrators.

* 1. **Transaction Requirements**

**2.2.1 Data Entry Requirements**

The data entry requirements of an active student management system outline the necessary functionalities and features to input, validate, and store student-related data accurately and efficiently. These requirements ensure that users can enter, update, and maintain student information effectively within the system. Here are the key data entry requirements for such a system:

1. **User-friendly Interface**:

- Intuitive and user-friendly interface for data entry to facilitate easy navigation and interaction with the system.

1. **Data Validation**:

- Validation of input data to ensure accuracy and integrity, including checks for data type, format, and completeness.

- Error messages and prompts to guide users in correcting invalid or missing information during data entry.

3. **Data Entry Forms**:

- Customizable data entry forms tailored to capture different types of student information, such as personal details etc.

- Dynamic forms that adapt based on the selected program, course, or student category to display relevant fields and options.

4. **Concurrency Control**:

- Implement concurrency control mechanisms to manage simultaneous data entry operations and prevent conflicts and inconsistencies.

- Use locking mechanisms, transaction isolation levels, and concurrency control

techniques to ensure data integrity and consistency in multi-user environments.

5. **Data Integrity Constraints**:

- Define and enforce data integrity constraints such as primary keys, foreign keys, unique constraints, and check constraints to ensure the validity and consistency of data entered into the database.

- Primary keys ensure uniqueness and identify each record uniquely, while foreign keys maintain referential integrity between related tables.

By incorporating these data entry requirements, the system can ensure accurate, efficient, and compliant handling of student data throughout its life-cycle within the organization.

**2.2.2 Data Retrieval Requirements**

The data retrieval requirements of an active student management system outline the functionalities and features necessary to retrieve, query, and analyze student-related data effectively. These requirements ensure that users can access relevant information efficiently to support decision-making, and academic planning processes. Here are the key data retrieval requirements for such a system:

1. **User-friendly Interface**:

- Intuitive and user-friendly interface for data retrieval to facilitate easy navigation and interaction with the system.

2. **Query and Search Capabilities**:

- Advanced query capabilities to search and retrieve student data based on various criteria such as student ID, name, enrollment status, program/course, academic year, etc.

- Support for complex queries, including filtering, sorting, and aggregation functions, to analyze student data and generate custom reports.

4. **Real-time Data Access**:

- Real-time access to student data to ensure that users can retrieve the most up-to-date information on student enrollment, academic performance, attendance, etc.

- Integration with other systems or data sources to synchronize and consolidate student data in real-time.

5. **Role-based Access Control**:

- Define access permissions to ensure that users can only retrieve data relevant to their roles and areas of responsibility.

6. **Data Aggregation and Analysis**:

- Aggregation and analysis capabilities to summarize and analyze student data across different dimensions (e.g., demographics, academic performance) for insights and decision-making.

- Support for data visualization techniques such as charts, graphs, and histograms to present data analysis results visually.

8. **Performance and Scalability**:

- Ensure optimal performance and scalability to handle large volumes of data and concurrent user requests for data retrieval.

- Implement caching mechanisms, query optimization techniques, and scalable infrastructure to improve response times and accommodate growing data loads.

By incorporating these data retrieval requirements, the system can provide users with the tools and capabilities they need to access, analyze, and utilize student data effectively for administrative, academic, and decision-making purposes.

**2.2.3 Data Updating Requirements**

The data updating requirements of an active student management system outline the functionalities and features necessary to modify, update, and maintain student-related data effectively. These requirements ensure that users can make changes to student records accurately and efficiently, reflecting any updates or modifications in the system. Here are the key data updating requirements for such a system:

1. **User Authentication and Authorization**:

- Authentication mechanisms to verify the identity of users accessing the system and ensure that only authorized personnel can perform data updates.

2. **Edit and Modify Real-time Student Records**:

- Ability to edit and modify existing student records to update personal information, contact details, enrollment status, academic program, etc.

- Provide user-friendly interfaces for data entry and editing, with intuitive forms and controls to facilitate easy updates.

1. **Data Validation and Error Handling**:

- Data validation mechanisms to validate updated information against predefined rules, formats, and constraints to ensure data integrity.

- Error handling mechanisms to detect and handle input errors or validation failures during data updating processes, providing appropriate feedback to users.

1. **Audit Trail and Logging**:

- Maintain an audit trail of data updating activities, recording details such as the user who made the update, the timestamp of the update, and the nature of the changes.

- Log data modification events to track changes made to student records for accountability, compliance, and auditing purposes.

1. **Concurrency Control**:

- Implement concurrency control mechanisms to manage simultaneous updates to student records and prevent conflicts or inconsistencies.

- Use locking mechanisms, transaction isolation levels, and optimistic or pessimistic locking techniques to ensure data integrity in multi-user environments.

By addressing these data updating requirements, an active student management system can support efficient and accurate updates to student records, ensuring that the information stored in the system remains current, reliable, and reflective of real-time changes in student information.

**2.2.4 Data Removal Requirements**

The data deleting requirements of an active student management system outline the functionalities and features necessary to delete student-related data effectively and securely. These requirements ensure that users can remove obsolete or unnecessary records from the system while maintaining data integrity. Here are the key data deleting requirements for such a system:

1. **User Authentication and Authorization**:

- Authentication mechanisms to verify the identity of users attempting to delete student data and ensure that only authorized personnel can perform deletion operations.

1. **Deletion Policies and Guidelines**:

- Define deletion policies and guidelines specifying under what circumstances student records can be deleted, such as upon graduation, withdrawal, or expiration of data retention periods.

1. **Data Purging and Archiving**:

- Implement data purging mechanisms to permanently delete student records from the system when they no longer serve any operational or legal purpose.

- Archive deleted records for a specified retention period before permanent deletion to comply with audit and regulatory requirements.

1. **Cascade Deletion and Referential Integrity**:

- Handle cascade deletion of related data to maintain referential integrity and prevent orphaned records when deleting student records associated with other entities (e.g., enrollment records, course registrations).

- Implement cascading deletion rules or foreign key constraints to automatically delete dependent records or update references when a parent record is deleted.

1. **Audit Trail and Logging**:

- Maintain an audit trail of deletion activities, recording details such as the user who initiated the deletion, the timestamp of the deletion, and the reason for deletion.

- Log deleted data and associated metadata to track changes made to student records and ensure accountability and transparency.

By addressing these data deleting requirements, The system can facilitate secure and compliant deletion of student records, supporting data governance, privacy, and risk management objectives while ensuring data integrity and accountability throughout the deletion life cycle.

1. **Database Design**

**3.1** **Conceptual database design of the new system**

This conceptual database design provides a foundation for capturing the essential data entities, attributes, relationships and their E-R diagram within the student management system at the medical campus.

**3.1.1 Entities with their description**

1. **Faculty:** This entity represents the teaching staff,principals and anyone who is not a student. Faculty members are responsible for delivering courses, providing academic guidance.
2. **Department**: refers to an organizational unit within the institution focused on a specific area of study or practice. Each department is typically led by a chairperson and comprises faculty members, researchers, and administrative staff who collaborate to advance medical knowledge, provide patient care, and educate students.
3. **Course**: refers to a specific educational program or module offered to students pursuing medical or healthcare-related degrees. These courses may cover a wide range of topics. Medical courses often include a combination of classroom lectures, laboratory work, clinical rotations, and hands-on training to provide students with a comprehensive understanding of the human body, diseases, and medical treatments.
4. **Student**: an individual who is enrolled in an educational program related to healthcare or medicine. This person is actively pursuing a degree or certification in fields such as medicine, nursing, pharmacy, dentistry, or other allied health professions. As a student, they are engaged in learning and developing the knowledge, skills, and competencies necessary to become a healthcare professional.
5. **Address**: a data structure used to identify and locate a specific place or location. It typically includes information such as the street name, house or building number, city, state or province, postal or ZIP code, and country.
6. **Tellno**: is a telephone number used to identify a specific person. It typically includes mobile numbers from the people.

**3.1.2 Attributes with their description**

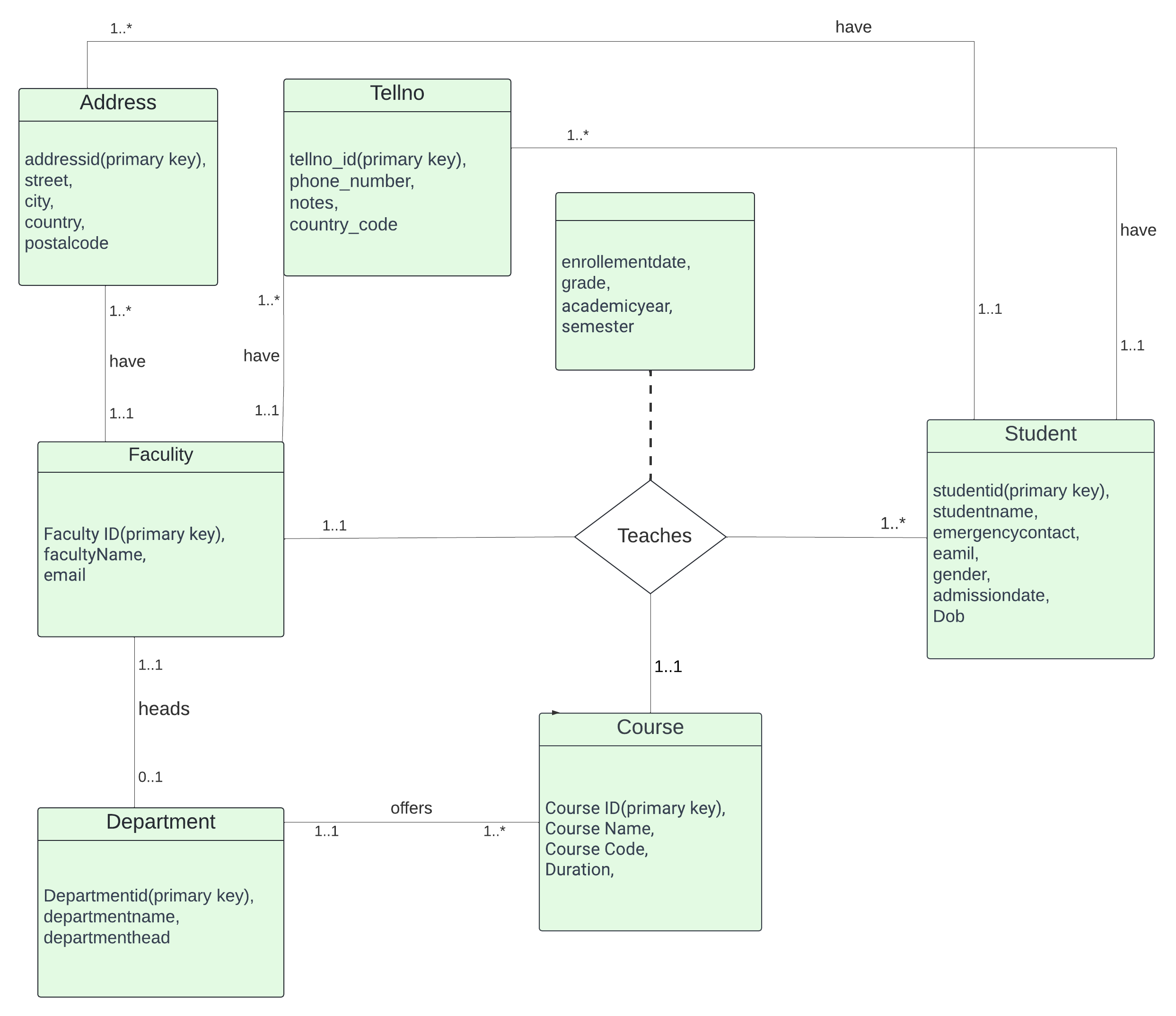
1. Faculty:
   1. Faculty\_id(primary key): distinct identification of faculty members.
   2. Faculty\_Fname: first name of faculty member.
   3. Faculty\_Lname: last name of faculty member.
   4. Email: email accounts of faculty members.
   5. Address: the address of a faculty member.
   6. Tell\_no: the phone number of a faculty member.
2. Courses:
   1. Course\_ID(primary key): distinct identification of courses.
   2. Course\_Name: name of the courses given.
   3. Course\_Code: the code for the given courses.
   4. Description: A brief description or overview of the course content.
   5. Duration: the time required to give the course.
3. Department:
   1. Department\_id(primary key): distinct identification of departments.
   2. Department\_name: the name of the department.
   3. Department\_head: the head of the specific department.
4. Student:
   1. Student\_id(primary key): distinct identification of students.
   2. Student\_Fname: first name of a student.
   3. Student\_Lname: last name of a student.
   4. Emergency\_contact: contact of a personal in case of emergency.
   5. Email: email of the student.
   6. Gender: gender of the student.
   7. Admission\_date: The specific calendar date when the student was admitted.
   8. Address: the address of a faculty member.
   9. Tell\_no: the phone number of a faculty member.
5. Address:
   1. Address\_id(primary key): distinct identification of the address.
   2. Street: the street number of the address.
   3. City: the city of the address.
   4. Country: the country of the address.
   5. Postal-code: the postal code for the address
6. Tell\_no:
   1. Tellno\_id(primary key): distinct identification of the telephone number.
   2. phone\_number: the phone number.
   3. country\_code: the country code of the telephone number.

**3.1.3 Relationships between the entities**

1. Faculty-course: one to one relationship as a faculty member can give one course and a course can only be given a faculty member.
2. Faculty-student: one to many relationship as a student can be lectured by one teacher and a teacher can lecture many students.
3. Course-student: many to many relationship as a student can take many courses and a course can be taken by many students.
4. Faculty-department: one to one relationship because a faculty member can head either one or no department but a department can only be headed by one faculty.
5. Department-course: one to many relationship as a department can have many courses but each course can only be given by one department.
6. Faculty-address: one to many relationship as a faculty member can have many addresses but an address can only be belong to one member.
7. Faculty-Tellno: one to many relationship as a faculty can have many phone numbers but a phone number can only belong to one faculty member.
8. Student-address: one to many relationship as a student can have many addresses but an address can only be belong to one student.
9. Student-Tellno: one to many relationship as a student can have many phone numbers but a phone number can only belong to one student.

3.1.3  **E-R diagram**

This E-R diagram is designed by lucid charts and captures the relationships between faculty, student, courses, address, tellno, department. The diagram uses standard UML notation to represent entities and their relationships.



**3.2 Logical Database Design**

**3.2.1 ER-Relation Mapping**

|  |  |
| --- | --- |
| **Faculty** (Faculty\_id, Faculty\_Fname, Faculty\_Lname, Email, Address, Tell\_no)  **Primary key**: Faculty\_id  **Foreign key**: address references Address(address\_id)  **Foreign key**: tellno references TellNo(tellno\_id) | **Courses** ( Course\_ID, Course\_Name,  Course\_Code, Duration,  Description,department\_id )  **Primary key**: Course\_id  **Foreign key**: department\_id references Department(Departmentid) |
| **Student** (Student\_id, Student\_Fname, Student\_Lname, DOB, Gender, Email,Tell\_no, ,Admission\_date, Address)  **Primary key**: student\_id  **Foreign key**: Address references Address(address\_id)  **Foreign key**: Tellno references TellNo(tellno\_id) | **Department** (department\_id, Department\_name,  Department\_head )  **Primary key**: department\_id  **Foreign key**: departmenthead references Faculty(Faculty\_id) |
| **Address** (addressid, street, city, country, postalcode)  **Primary key** : addressid | **Class** (faculty\_id, course\_id, student\_id, enrollementdate, grade, semester, academicyear)  **Primary key**: course\_id, student\_id  **Foreign key**: faculty\_id references Faculty(faculty\_id)  **Foreign key**: course\_id references Course(course\_id)  **Foreign key**: student\_id references Student(student\_id) |
|  | **TellNo** (tellno\_id, phone\_number, notes,  country\_code)  **Primary key**: tellno\_id |

3.2.2 **Validating model with Normalization**

### 3.2.2.1 **First Normal Form**

The first normal form requires that each table has no repeating groups and that every attribute value should be atomic and no repeating groups should be allowed. Based on these conditions, we will check to see if the tables satisfy the 1NF.

i) The **Faculty** table: satisfies the 1NF rule as each column has an atomic value.

ii) The **Courses** table: satisfies the 1NF rule as each column has an atomic value.

iii) The **Student** table: satisfies the 1NF rule as each column has an atomic value.

iv) The **Department** table: satisfies the 1NF rule as each column has an atomic value.

v) The **Class** table: satisfies the 1NF rule as each column has an atomic value.

vi) The **Address** table: satisfies the 1NF rule as each column has an atomic value.

vii) The **TellNo** table: satisfies the 1NF rule as each column has an atomic value

### 3.2.2.2 Second Normal Form

Requires for all non-key attributes to be fully dependent on the primary key; no partial dependencies should be allowed. And also the table must be in the first normal form.

NB: if a table is in first normal form and its primary key consists of only one attribute the table is also automatically in second normal form because no partial dependencies are possible. in this case all of the tables in the database qualify as second normal form automatically except for the class table. So we should inspect it further.

**Class** ( course\_id, student\_id, enrolementdate, faculty\_id, grade, semester, academicyear)

Functional dependencies in the class table are the following:

FD1: course\_id & student\_id →enrolment date , grade

FD2: course\_id→ academic year, semester, faculty\_id

Based on these functional dependencies the three attributes listed in FD2 are partially dependent on the key attributes so we create a new table for them.

|  |  |  |  |
| --- | --- | --- | --- |
| Class student | | | |
| ***course\_id*** | ***Student\_id*** | Enrolment date | grade |
| Class year | | | |
| ***course\_id*** | Academic year | semester | Faculty\_id |

### 3.2.2.3 **Third Normal Form**

The third normal form requires for there to not be any transitive dependencies between non-key attributes and must be on the first and second normal form.

A transitive dependency exists when a non-key attribute depends on another non-key attribute. All the tables in the database are in the second normal form so we look for transitive dependencies.

All the non-key attributes in all the tables are fully dependent on the primary key and nothing but the key so the tables are also in third normal form.

## 3.2.3 Relational Schema with Referential Integrity after Normalization

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Faculty** | | | | | | | | |
| Faculty  id | Faculty  Fname | Faculty\_  Lname | Email | Position | Gender | Address | Tel  no | Department  id |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Courses** | | | | |
| Course\_ID | Course\_Name | Course\_Code | Duration | Description |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Student** | | | | | | | | |
| Student  id | Student  Fname | Student  Lname | Gender | Emergency  contact | Tel  no | Email | Admission  date | Address |

|  |  |  |
| --- | --- | --- |
| **Department** | | |
| department\_id | Department\_name | Department\_head |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | | | | |
| addressid | street | city | country | postalcode |

|  |  |  |  |
| --- | --- | --- | --- |
| **TellNo** | | | |
| tellno\_id | phone\_number | notes | country\_code |

|  |  |  |  |
| --- | --- | --- | --- |
| Class student | | | |
| ***course\_id*** | ***Student\_id*** | Enrolment date | grade |

|  |  |  |  |
| --- | --- | --- | --- |
| Class year | | | |
| ***course\_id*** | Academic year | semester | Faculty\_id |

**3.3 Physical database design**

**3.2.3 Physical design strategy**

The physical design strategy for active student management system database will be tailored to the specific requirements and constraints of the system, considering available resources and technologies. Here are the guidelines we have followed:

1. **Identify Entities and Relationships**: The initial step involves identifying the entities and relationships relevant to student management. This includes student information, courses, instructors, and other pertinent entities.

2. **Normalize the Data**: To ensure data consistency and reduce redundancy, the next step is normalizing the data. This process involves identifying functional dependencies between attributes and organizing them into tables, adhering to normalization principles.

3. **Determine Data Types and Constraints**: Once tables are identified, determine appropriate data types and constraints for each column. This includes specifying data range, data type, and unique constraints to maintain data integrity.

4. **Define Indexing Strategy**: Improve query performance by determining the indexing strategy. The choice of indexing will depend on the nature and volume of data within the student management system.

5. **Establish Storage Strategy**: Determine how data is stored on disk, considering factors such as file organization method, allocation strategy, and storage structure. This step aims to optimize data retrieval and storage efficiency.

6. **Address Security and Access Control**: Implement security measures to protect sensitive student data. This involves encrypting data and restricting access based on user roles and privileges to ensure confidentiality and integrity.

7. **Plan for Scalability and Availability**: Lastly, plan for the system's scalability and availability to accommodate increasing volumes of student data and traffic. This may involve implementing clustering or replication technologies to distribute data and maintain system load balance.

**3.2.3 Database Deployment details**

In certain cases, certain Database Management Systems (DBMSs) may automatically determine the optimal file organization strategy without allowing explicit selection by the user. For our specific system, we will be using Microsoft SQL Server, which inherently supports B+ tree file organization. Consequently, we will utilize the B+ tree file organization method.

To select appropriate indexing strategies, we initially identified the attributes that are frequently used to access tuples within each relation.

First we listed the attributes that are used most often to access tuples in relation.

* In the student relation the studentid is used more often to access the tuples in that relation.
* In the department relation the departemntID is used more often to access the tuples in that relation.
* In the course relation the courseID is used more often to access the tuples in that relation.
* In the address relation the addressID and studenId is used more often to access the tuples in that relation.
* In the Tellno relation the tellId and studentid is used more often to access the tuples in that relation.
* In the Faculty relation the FacultyID is used more often to access the tuples in that relation.

To summarize, we will apply primary indexing to the specified attributes since they serve as keys within their respective relations. Additionally, we have chosen not to use secondary indexing for most relations due to their small size. because none of the other attributes in that relation are commonly used for tuple access.

To estimate the disk storage requirements for each table, we have compiled a list of relevant variables or attributes associated with each table.

**Student**

Number of Attributes: 6

Size of bytes reserved for each attribute: determined as the datatype.

**Address**

Number of Attributes: 5

Size of bytes reserved for each attribute: determined as the datatype

**TellNo**

Number of Attributes: 4

Size of bytes reserved for each attribute: determined as the datatype

**Department**

Number of Attributes: 3

Size of bytes reserved for each attribute: determined as the datatype

**Faculty**

Number of Attributes: 4

Size of bytes reserved for each attribute: determined as the datatype

**Course**

Number of Attributes: 5

Size of bytes reserved for each attribute: determined as the datatype

### Hardware Implementation

|  |  |
| --- | --- |
| Processor | Intel core i5 processor |
| Operating system | Window 10 |
| Memory | 8GB RAM |
| Internal storage space | 1000GB |
| DBMS | SQL SERVER |
| Networking Device | WIFI |

**Installation and configuration of the database software**

Among well-known options like Oracle, Microsoft SQL Server, and MySQL, we’ve opted for Microsoft SQL Server for our actual implementation.

SQL Server is a robust **Relational Database Management System (RDBMS)**.

**Primary Functions**:

* **Storage and Access**: SQL Server efficiently stores and retrieves data as required by various applications.
* **Data Integrity**: It ensures data consistency, security, and reliability.
* **Scalability**: SQL Server can handle large-scale databases and high transaction volumes
* **Business Intelligence**: It supports data warehousing, reporting, and analytics.
* **Programming Languages**: SQL Server supports languages like C++, Java, Ruby, Visual Basic, Delphi, and more.
* **Backup**: It doesn’t block the database during data backup.
* **Editions**: Available in various editions (Enterprise, Standard, Web, Workgroup, Express).

1. **Implementation and testing**
   1. **SQL script for creating the database**

create database active\_student\_mangement\_system;

* 1. **SQL Scripts for creating the tables, view, indexes**

----------Tables

Create table Adress(

Addressid varchar(50) primary key,

Street varchar(50),

City varchar(50),

Country varchar(50),

Postalcode int

)

Create table tellNo(

Tellno\_id varchar(50) primary key,

Phone\_number int,

Notes varchar (50),

Country\_code int

)

create table student( --------student

studentid varchar(50) primary key,

studentFname varchar(50),

studentLname varchar(50),

email varchar(50),

gender varchar(10),

DOB date,

admissiondate date,

saddress varchar(50),

stellNo varchar(50),

foreign key(saddress) references Adress(Addressid),

foreign key(stellNo) references tellNo(Tellno\_id)

)

create table faculty( ------faculty

facultyid varchar(50) primary key,

facultyFname varchar(50),

facultyLname varchar(50),

email varchar(50),

faddress varchar(50),

ftellNo varchar(50),

foreign key(faddress) references Adress(Addressid),

foreign key(ftellNo) references tellNo(Tellno\_id)

)

create table department( ------department

departmentid varchar(50) primary key,

departmentname varchar(50),

department\_head varchar(50),

foreign key(department\_head) references faculty(facultyid)

)

create table course( ------course

courseid varchar(50) primary key,

coursename varchar(50),

coursecode varchar(50),

descriptions varchar(50),

duration varchar(50),

department varchar(50),

foreign key(department) references department(departmentid)

)

Create table classtudent (

course\_id varchar(50),

student\_id varchar(50),

Enrollementdate date ,

Grade varchar(5),

Primary key(course\_id, student\_id),

Foreign key(course\_id) references Course(courseid),

Foreign key(student\_id) references Student(studentid)

)

create table classyear(

course\_id varchar(50),

Semester int,

Academicyear date,

faculty\_id varchar(50),

Primary key(course\_id),

Foreign key(faculty\_id) references Faculty(facultyid),

Foreign key(course\_id) references Course(courseid),

)

----------indexes

Create index faculty\_idx

On faculty(facultyFname);

Create index student\_idx

On student(studentFname);

Create index address\_idx

On Adress (city);

Create index tellno\_idx

On tellno (phone\_number);

Create index class\_student\_idx

On classtudent (grade);

Create index class\_year\_idx

On classyear (academicyear);

------------views

CREATE VIEW student\_view AS

SELECT student. studentFname, student.studentLname, classtudent.grade

FROM student

JOIN classtudent ON student. studentid= classtudent.student\_id;

CREATE VIEW faculty\_view AS

SELECT faculty.facultyFname, faculty.facultyLname, faculty.faddress

FROM faculty;

* 1. **Testing (with sample data-for all CRUD operations)**

insert into Adress

values('1','gulele','addis ababa','ethiopia','1500'),

('2','kaliti','addis ababa','ethiopia','1100'),

('3','bole','addis ababa','ethiopia','1000'),

('4','arada','addis ababa','ethiopia','1000'),

('5','kolfe','addis ababa','ethiopia','1000'),

('6','kirkos','addis ababa','ethiopia','1900'),

('7','lideta','addis ababa','ethiopia','1200'),

('8','shiro meda','addis ababa','ethiopia','1200')

--

insert into tellNo(Tellno\_id,Phone\_number,Country\_code)

values('1','0911223344','+251'),

('2','0922331144','+251'),

('3','0922113344','+251'),

('4','0921123344','+251'),

('5','0923341124','+251'),

('6','0923123414','+251'),

('7','0912233445','+251'),

('8','0978455634','+251')

insert into student

values('001','betelehem','ayele','abebekebede@gmail.com','female','2002-02-08','2022-12-12','2','3'),

('002','tedios','manachew','tediosmanachew@gmail.com','male','1985-12-28','2012-02-02','1','1'),

('003','xilahun','haile','xilahunhaile@gmail.com','male','1978-05-14','2015-12-12','3','2'),

('004','tola','bele','tolabele@gmail.com','male','1999-12-09','2018-12-18','4','4')

insert into faculty

values('101','kebede','aleme','kebedealemu@gmail.com','6','5'),

('102','tola','dereje','toladereje@gmail.com','5','6'),

('103','zewedu','melaku','zewedumelaku@gmail.com','7','8'),

('104','sena','almayehu','senaalemayehu@gmail.com','8','7')

insert into department

values('N1001','Anstesia','101'),

('N1012','Radiology','103'),

('N1005','Nursing and midwifery','104')

insert into course

values('c001','anatomy','Ac001','the anatomy of the body','3 month','N1005'),

('c002','biology','Rc001','living and non living things','3 month','N1012'),

('c003','micro biology','Ac003','micro organisms on the body','3 month','N1005'),

('c004','pharmacology','Ac003','the medicinal knowledge for anesthesia','3 month','N1001')

insert into classtudent

values('c002', '001', '2023-07-10', 'A'),

('c003', '004','2023-02-10','B'),

('c002', '002','2023-02-10','A'),

('c001', '003','2023-07-10','A');

INSERT INTO classyear

VALUES

('c002','2', '1','101'),

('c003','1', '1','102'),

('c002','1', '1','101'),

('c001','2', '2','103');

1. **References**

Fundamentals\_of\_Database\_Systems\_Elmasri\_and\_Navathe\_7th\_edition.

SQL For Dummies" by Allen G. Taylor, "Learning SQL" by Alan Beaulieuc ,An Introduction to Database Systems.

8th edn. Reading, MA: Addison-Wesley.

Date C.J. (2012). Database Design and Relational Theory: Normal Forms and All That Jazz. 1st edn.

O’Reilly Media. Github ,SQL Server Management Studio (SSMS), lucid chart.

1. **Appendix(Forms, Reports of the organization etc.)**

In this appendix, we will provide examples of forms and reports commonly used in the organization. These forms and reports play a crucial role in managing student data, tracking academic progress, and ensuring compliance with regulatory requirements.

1. Forms:

- Student Enrollment Form: Used to collect basic information from students when they enroll in the medical school, including personal details, contact information, educational background, and emergency contacts.

- Course Registration Form: Allows students to enroll in courses for each semester, indicating their preferred schedule and elective choices.

2. Reports:

- Student Progress Report: Provides an overview of each student's academic performance, including grades, attendance records, clinical rotation evaluations, and overall progress towards graduation.

- Course Evaluation Report: Summarizes feedback from students on each course, including ratings on teaching quality, course content, learning outcomes, and suggestions for improvement.

- Graduation Rate Report: Tracks the percentage of students who successfully complete the program within a specified timeframe, highlighting graduation rates by cohort and demographic factors.

These forms and reports are essential tools for medical school administrators, faculty members, and students to facilitate communication, monitor progress, and ensure accountability in academic processes. By utilizing standardized forms and generating informative reports, medical schools can enhance transparency, efficiency, and effectiveness in managing student data and organizational operations.