**Case Scenario: Coursework Tracking System**

1. **References:**
2. Journal Article:
   * Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7th ed.). Pearson.
     + This book provides a comprehensive overview of database design principles, including conceptual and logical design, which is highly relevant to the coursework tracking system.
3. Book Chapters:
   * Connolly, T., & Begg, C. (2014). *Database Systems: A Practical Approach to Design, Implementation, and Management* (6th ed.). Pearson.
     + Chapters 16 and 17 focus on database design methodologies and normalization, which are essential for this task.
   * Hoffer, J. A., Ramesh, V., & Topi, H. (2016). *Modern Database Management* (12th ed.). Pearson.
     + Chapter 5 discusses entity-relationship modeling, which is critical for conceptual schema design.
4. Websites:
   * Lucidchart. (n.d.). *What is an Entity Relationship Diagram (ERD)?*  
     Available at: <https://www.lucidchart.com/pages/er-diagrams>
     + This resource explains ER diagrams and their use in database design.
   * MySQL Workbench. (n.d.). *Database Design and Modeling*.  
     Available at: <https://www.mysql.com/products/workbench/>
     + This website provides tools and tutorials for logical database design using MySQL Workbench.

# **b. Requirements Analysis**

By reading the case scenario, I have examined the essential components of the Coursework Tracking System (CTS) .By checking who will be using the system, what information it will hold, to make it user-friendly to access any data

**Users and Their Roles**

**Students**, **instructors**, and **system administrators** are the three primary user categories that the CTS is intended to serve.

The system must include to the following users and their requirements:

1. **Students:**
   * Track assignments, deadlines, and submission status.
   * View course details and instructor information.
   * Set reminders for upcoming deadlines.
2. **Instructors:**
   * Add course and assignment details.
   * Update assignment deadlines and requirements.
3. **Administrators:**
   * Manage user accounts (students and instructors).
   * Maintain course and assignment records.

**Functional Requirements:**

* Store details of courses, assignments, instructors, and students.
* Allow students to view and update their assignment status.
* Provide reminders for upcoming deadlines.
* Ensure data integrity and avoid redundancy.

**Non-Functional Requirements:**

* Scalability to handle multiple users.
* Security to protect user data.
* Efficient data retrieval and storage.

**c. Conceptual Schema Design:**

Using an Entity-Relationship Diagram (ERD), the following entities and relationships are identified:

**Entities:**

1. **Student:**
   * Attributes: StudentID (PK), FirstName, MiddleName, LastName, Email, Password.
2. **Course:**
   * Attributes: CourseID (PK), CourseName, Season, Year, Credits, InstructorID(FK),
3. **Instructor:**
   * Attributes: InstructorID (PK), FirstName, MiddleName, LastName, Email, Password
4. **Assignment:**
   * Attributes: AssignmentID (PK), Title, Description, Deadline, CourseID (FK)
5. **Reminder:**
   * Attributes: ReminderID (PK), AssignmentID (FK), ReminderDateTime, Message

**Relationships:**

* A **Student** enrolls in multiple **Courses**. A **Course** can be taken by Many **Students** (Many-to-Many).
* A **Course** is taught by one **Instructor** , but one **Instructor** can teach many **Courses** (One-to-Many).
* A **Student** takes multiple **Assignments**. An **Assignment** can be given to many **Students**. (Many-to-Many)
* An **Assignment** belongs to one **Course** but one **Course** can have many **Assignments** (One-to-Many).
* A **Reminder** is associated with one **Assignment** but an **Assignment** can have many **Reminders** (One-to-Many).

**Logical Design:**

The logical design involves mapping the conceptual schema to a relational database model.

This includes:

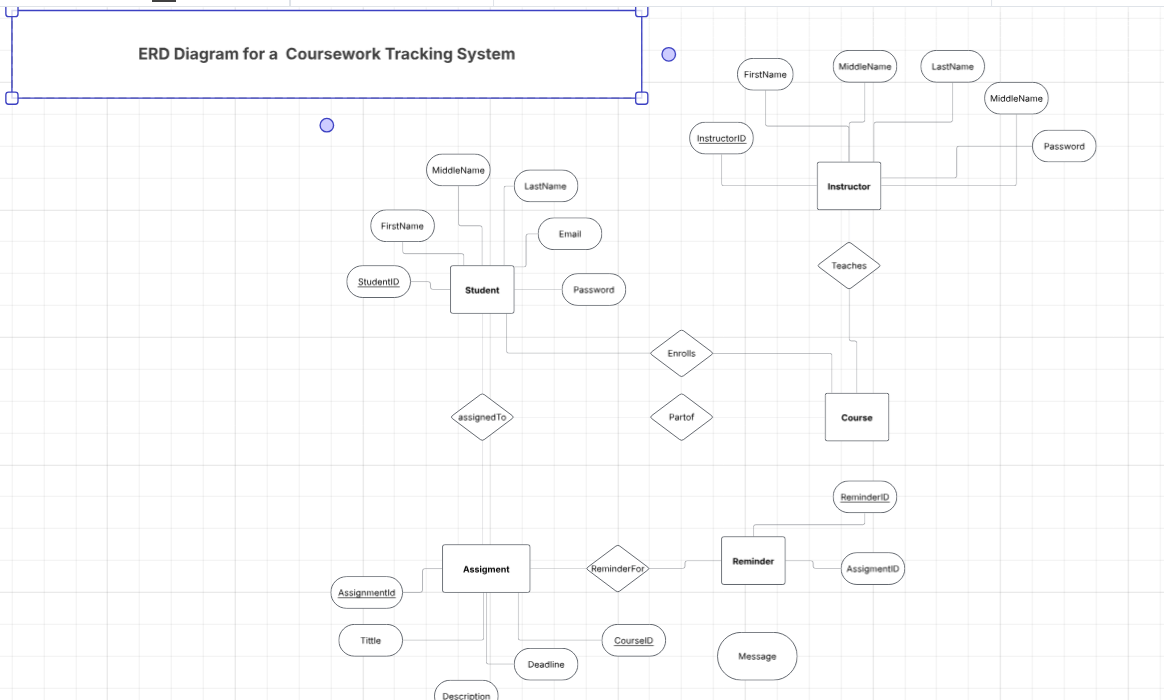
**Tables:**

* + **Student** (StudentID(PK), FirstName, MiddleName, LastName, Email, Password).
  + **Course** (CourseID(PK), CourseName, Credits, InstructorID(FK)).
  + **Instructor** (InstructorID(PK), FirstName, MiddleName, LastName, Email, Password).
  + **Assignment** (AssignmentID(PK), Title, Description, Deadline, CourseID(FK)).
  + **Reminder** (ReminderID(PK), AssignmentID(FK), ReminderDateTime, Message).
  + **Enrollment** (StudentID, CourseID) – for the Many-to-Many relationship between Student and Course. Here, the Primary Key is StudentID, CourseID together.
  + **StudentAssignmentRecords**(StudentID, AssignmentID) - Here Primary Key is StudentID, AssignmentID together.

1. **Primary and Foreign Keys:**
   * Primary keys are represented by PK.
   * Foreign keys are represented by FK.
2. **Constraints:**
   * NOT NULL, UNIQUE, and CHECK constraints for data integrity.

A diagram of a student

AI-generated content may be incorrect.

. 

**Task 2:**

**a. References:**

1. **Journal Article:**
   * Date, C. J. (2003). *An Introduction to Database Systems* (8th ed.). Addison-Wesley.
     + This book discusses normalization and anomaly resolution in database design.
2. **Book Chapters:**
   * Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). *Database Systems: The Complete Book* (2nd ed.). Pearson.
     + Chapter 3 covers normalization and anomaly identification.
   * Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). *Database System Concepts* (6th ed.). McGraw-Hill.
     + Chapter 7 discusses performance optimization techniques.
3. **Websites:**
   * GeeksforGeeks. (n.d.). *Normalization in DBMS*.  
     Available at: <https://www.geeksforgeeks.org/normalization-in-dbms/>
     + This resource explains normalization and its role in eliminating anomalies.
   * MySQL. (n.d.). *Optimization and Indexing*.  
     Available at: <https://dev.mysql.com/doc/refman/8.0/en/optimization.html>
     + This website provides guidelines for optimizing MySQL databases.

**b. Issues with Logical Design:**

As we can observe there are no Anomalies that can be identified here:

**Student Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **StudentID** | **FirstName** | **MiddleName** | **LastName** | **Email** | **Password** |
| S001 | John | A. | Doe | johndoe@gmail.com | pass123 |
| S002 | Jane | B. | Smith | [janesmth@gmail.com](mailto:janesmth@gmail.com) | pass456 |
| S003 | David | C. | Miller | davidmill@gmail.com | pass789 |

**Instructor Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **InstructorID** | **FirstName** | **MiddleName** | **LastName** | **Email** | **Password** |
| I001 | Alice | C. | Johnson | [alice@univ.edu](mailto:alice@univ.edu) | teach123 |
| I002 | Bob | (null) | Williams | [bob@univ.edu](mailto:bob@univ.edu) | teach456 |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | | |  |  |
| **Course Tbale** |  | | |  |  |
| **CourseID** | **CourseName** | | | **Credits** | **InstructorID** |
| C101 | Intro to Programming | | | 3 | I001 |
| C102 | Database Systems | | | 4 | I002 |
| **Assignment Table** |  | | |  |  |
| **AssignmentID** | | **Title** | **Description** | | | | **Deadline** | **CourseID** |
| A001 | | Homework #1 | Basic programming problems | | | | 31/03/25 23:59 | C101 |
| A002 | | Homework #2 | Database normalization exercise | | | | 01/04/25 23:59 | C102 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reminder Table**  **ReminderID** | **AssignmentID** | **ReminderDateTime** | **Message** |  |
| R001 | A001 | 25/03/25 10:00 | Finish HW1 a week early! | |
| R002 | A001 | 30/03/25 18:00 | Submit HW1 by midnight! | |
| R003 | A002 | 28/03/25 9:00 | Start working on DB exercise early | |

|  |  |
| --- | --- |
| **Enrollment** |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **StudentID** | **CourseID** |
| S001 | C101 |
| S001 | C102 |
| S002 | C102 |
| S003 | C101 |
|  |  |

|  |  |
| --- | --- |
| **StudentAssignmentRecords** | |
|  |  |
| **StudentID** | **AssignmentID** |
| S001 | A001 |
| S001 | A002 |
| S002 | A002 |
| S003 | A001 |

1. **Insert Anomaly:**
   * No Insertion Anomalies found since tables are designed properly and are in **3NF**
2. **Update Anomaly:**
   * No Update Anomalies found since tables are designed properly and are in **3NF**
3. **Delete Anomaly:**
   * No Delete Anomalies found since tables are designed properly and are in **3NF**

**c. Performance Optimization:**

1. **Normalization:**

We ensure 3NF to eliminate redundancy and anomalies.

**Student** (StudentID (PK), FirstName, MiddleName, LastName, Email, Password)

Primary Key is StudentID

(Functional dependency we have is StudentID -> FirstName, MiddleName, LastName, Email, Password)

There are no partial dependencies.

There are no transitive dependencies.

Therefore the **Student** table is in **3NF**

**Course** (CourseID(PK), CourseName, Credits, InstructorID(FK)).

Primary Key is CourseID

(Functional dependency we have is CourseID -> CourseName, Credits, InstructorID)

There are no partial dependencies.

There are no transitive dependencies.

Therefore the **Course** table is in **3NF\**

**Instructor** (InstructorID(PK), FirstName, MiddleName, LastName, Email, Password).

Primary Key is InstructorID

Functional dependency we have is InstructorID -> FirstName, MiddleName, LastName, Email, Password

There are no partial dependencies.

There are no transitive dependencies.

Therefore, the **Instructor** table is in **3NF**.

**Assignment** (AssignmentID(PK), Title, Description, Deadline, CourseID(PK)).

Primary Key: AssignmentID.

Functional dependency we have is AssignmentID -> Title, Description, Deadline, CourseID.

No partial dependency (single-attribute PK).

No transitive dependency.

This table is in **3NF**.

**Reminder** (ReminderID(PK), AssignmentID(FK), ReminderDateTime, Message).

Primary Key: ReminderID.

Functional dependency we have is ReminderID -> AssignmentID, ReminderDateTime, Message.

No partial dependency (single-attribute PK).

No transitive dependency.

Conclusion: This table is in **3NF**.

**Enrollment** (StudentID, CourseID)

Primary Key here is the Composite key (StudentID, CourseID).

Because there are no non-key attributes, there can't be partial dependencies or transitive dependencies.

Conclusion: This table is in **3NF**

**StudentAssignmentRecords**(StudentID, AssignmentID) Here Primary Key is StudentID, AssignmentID together.

Primary Key here is Composite key (StudentID, AssignmentID).

Similarly to Enrollment, there are no non-key attributes to violate normal forms.

Conclusion: This table is in **3NF** as well

1. **Indexing:**
   * Create indexes on **frequently** queried columns (e.g., StudentID, CourseID).
2. **Caching:**
   * Use caching mechanisms for frequently accessed data (e.g., course details).

**: Personal Reflection**

**Learning:**

Designing the **Coursework Tracking System** database was a good experience that really helped me to apply theoretical database understanding into a practical scenario.

1. **Conceptual Design:**
   * Converting the required case scenario into an Entity-Relationship Diagram (ERD) was an important skill I developed. I learned how to identify entities, their attributes, and the relationships between them.
   * Recognizing the Many-to-Many relationship between **Student** and **Course** help me to create the tables.
2. **Logical Design:**
   * Mapping the ERD to a relational schema not only help with my understanding but also allowed me to experience how data is kept and used by its user.
   * I came to know of normalization, which helped to see any redundancy and anomalies in the database.
3. **Using Tools for Data Base:**
   * Using tools like Lucid chart for ERD creation and MySQL Workbench for logical design enhanced my technical skills and gave me the confidence in working with database design software.

**Challenges Faced:**

**Identifying Anomalies:**

* + Initially, I found difficulty to spot insertion, update, and deletion anomalies in the logical design. For example, I didn’t realize that deleting a student record could also remove their course enrolment data.
  + **Solution:** I addressed this by applying normalization techniques, making sure that the database to the Third Normal Form (3NF). This helped in redundancy and resolved anomalies.

**How Challenges Were Addressed:**

1. **Normalization:**
   * By breaking down tables and ensuring each table showed a single purpose, I discarded the redundancy and resolved anomalies.
   * **Evidence:** This is shown in the logical design section of the report.
2. **Research and Justification:**
   * For performance optimization, I used the resources provided on the online library and online videos to make productive decisions.

**Evidence of Learning:**

1. **Conceptual Design:**
   * The ERD created using Lucid chart helped me to identify entities, attributes, and relationships.
2. **Logical Design:**
   * The relational schema and annotated screenshots in the assignment show how I linked the ERD to a logical design using MySQL Workbench. The video is used to explain.
3. **Anomaly Resolution:**
   * The table in the report highlighting insertion, update, and deletion anomalies, along with the steps taken to resolve them, provides understanding of normalization.
4. **Performance Optimization:**
   * The report includes a indexing, partitioning, and caching, supported by references to academic and online resources.

**Conclusion:**

This project was a very good experience that helped my understanding of database design principles. While going through some challenges, I overcame them through some research, practice, and deep thinking. The evidence provided in the report, including the ERD, logical schema, and video demonstration, shows my ability to design and a functional database system. This experience has helped me for more complex database design tasks in the future.