Design DB model for Guvi Zen class

1. Requirements for Guvi Zen Class

- Users : Students and instructors.
- Courses : Each course may have multiple modules.
- Enrolment : Students can enrol in multiple courses.
- Progress Tracking : Track student progress within courses.
- Assignments : Courses may have assignments or quizzes.
- Feedback : Students can provide feedback on courses.
- Certificates : Students can receive certificates upon course completion.

2. Identify Entities

1. User : Represents both students and instructors.
2. Course : Represents a learning module.
3. Module : Subdivision of a course.
4. Enrolment : Tracks student enrolment in courses.
5. Assignment : Contains details about assignments or quizzes.
6. Feedback : Feedback given by students on courses.
7. Certificate : Certificate issued upon course completion.

3. Identify Attributes

1. User
   - `user_id` (Primary Key)
   - `username`
   - `password`
   - `email`
   - `role` (e.g., student, instructor)
   - `full_name`
   - `date_of_birth`
   - `registration_date`

2. Course
   - `course_id` (Primary Key)
   - `title`
   - `description`
   - `instructor_id` (Foreign Key from User)
   - `creation_date`
   - `duration` (e.g., number of weeks)
   - `category`

3. Module
   - `module_id` (Primary Key)
   - `course_id` (Foreign Key from Course)
   - `title`
   - `description`
   - `sequence_number`

4. Enrollment
   - `enrollment_id` (Primary Key)
   - `user_id` (Foreign Key from User)
   - `course_id` (Foreign Key from Course)
   - `enrollment_date`
   - `progress_percentage`

5. Assignment
   - `assignment_id` (Primary Key)
   - `course_id` (Foreign Key from Course)
   - `title`
   - `description`
   - `due_date`

6. Feedback
   - `feedback_id` (Primary Key)
   - `course_id` (Foreign Key from Course)
   - `user_id` (Foreign Key from User)
   - `rating` (e.g., 1 to 5 stars)
   - `comment`
   - `feedback_date`

7. Certificate
   - `certificate_id` (Primary Key)
   - `user_id` (Foreign Key from User)
   - `course_id` (Foreign Key from Course)
   - `issue_date`
   - `expiry_date`

4. Identify Relationships

1. User to Course :
   - Many-to-Many relationship via `Enrollment`
   - A `User` can enroll in many `Courses`, and a `Course` can have many `Users`.

2. Course to Module :
   - One-to-Many relationship
   - A `Course` can have multiple `Modules`.

3. Course to Assignment :
   - One-to-Many relationship
   - A `Course` can have multiple `Assignments`.

4. Course to Feedback :
   - One-to-Many relationship

- A `Course` can have multiple `Feedback` entries.

5. User to Feedback :
- One-to-Many relationship
- A `User` can provide multiple `Feedback` entries.

6. User to Certificate :
- One-to-Many relationship
- A `User` can have multiple `Certificates`.

5. Create Tables

```sql
CREATE TABLE User (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    role ENUM('student', 'instructor') NOT NULL,
    full_name VARCHAR(100),
    date_of_birth DATE,
    registration_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Course (
    course_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    description TEXT,
    instructor_id INT,
    creation_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    duration INT,
    category VARCHAR(50),
```

```
    FOREIGN KEY (instructor_id) REFERENCES User(user_id)
);

CREATE TABLE Module (
    module_id INT AUTO_INCREMENT PRIMARY KEY,
    course_id INT,
    title VARCHAR(100) NOT NULL,
    description TEXT,
    sequence_number INT NOT NULL,
    FOREIGN KEY (course_id) REFERENCES Course(course_id)
);

CREATE TABLE Enrollment (
    enrollment_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    course_id INT,
    enrollment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    progress_percentage DECIMAL(5,2),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (course_id) REFERENCES Course(course_id)
);

CREATE TABLE Assignment (
    assignment_id INT AUTO_INCREMENT PRIMARY KEY,
    course_id INT,
    title VARCHAR(100) NOT NULL,
    description TEXT,
    due_date DATE,
    FOREIGN KEY (course_id) REFERENCES Course(course_id)
);
```

```
CREATE TABLE Feedback (
    feedback_id INT AUTO_INCREMENT PRIMARY KEY,
    course_id INT,
    user_id INT,
    rating INT CHECK(rating BETWEEN 1 AND 5),
    comment TEXT,
    feedback_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (course_id) REFERENCES Course(course_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);

CREATE TABLE Certificate (
    certificate_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    course_id INT,
    issue_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expiry_date DATE,
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (course_id) REFERENCES Course(course_id)
);
```