DATABASE MODEL

Creation script:

```
-- USERS TABLE
CREATE TABLE users (
   id BIGINT PRIMARY KEY IDENTITY(1,1),
   name NVARCHAR(100) NOT NULL,
   email NVARCHAR(100) UNIQUE NOT NULL,
   password_hash NVARCHAR(255) NOT NULL,
   created_at DATETIME2 DEFAULT GETDATE(),
   updated_at DATETIME2 DEFAULT GETDATE()
);

-- COURSES TABLE
CREATE TABLE courses (
   id BIGINT PRIMARY KEY IDENTITY(1,1),
   title NVARCHAR(150) NOT NULL,
   description NVARCHAR(1000),
   created_at DATETIME2 DEFAULT GETDATE(),
   updated_at DATETIME2 DEFAULT GETDATE()
);

-- JUNCTION TABLE: USER_COURSES (Many-to-Many)
CREATE TABLE user_courses (
   user_id BIGINT NOT NULL,
   course_id BIGINT NOT NULL,
   enrolled_at DATETIME2 DEFAULT GETDATE(),
   PRIMARY KEY (user_id, course_id),
   FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
   FOREIGN KEY (course_id) REFERENCES courses(id) ON DELETE CASCADE
);
```

-- ASSIGNMENTS TABLE

```
CREATE TABLE assignments (
   id BIGINT PRIMARY KEY IDENTITY(1,1),
   course_id BIGINT NOT NULL,
   title NVARCHAR(200) NOT NULL,
   description NVARCHAR(2000),
   deadline DATE NOT NULL,
   created_at DATETIME2 DEFAULT GETDATE(),
   updated_at DATETIME2 DEFAULT GETDATE(),
   FOREIGN KEY (course_id) REFERENCES courses(id) ON DELETE CASCADE
);
```

-- **TASKS TABLE**

```
CREATE TABLE tasks (
   id BIGINT PRIMARY KEY IDENTITY(1,1),
   assignment_id BIGINT NOT NULL,
   user_id BIGINT NOT NULL, -- Made NOT NULL: every task must belong to a user
   description NVARCHAR(1000) NOT NULL,
   done BIT DEFAULT 0,
   created_at DATETIME2 DEFAULT GETDATE(),
   updated_at DATETIME2 DEFAULT GETDATE(),
   FOREIGN KEY (assignment_id) REFERENCES assignments(id) ON DELETE CASCADE,
   FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Relations:

Users table

- Represents students (or potentially also instructors).
- **Primary key:** id
- **Relationships:**
    - Has a **many-to-many** relationship with courses via the user_courses table.
    - Has a **one-to-many** relationship with tasks (when user_id is not null), meaning a user can have multiple personal tasks.

Courses table

- Represents a university course.
- **Primary key:** id
- **Relationships:**
    - Has a **many-to-many** relationship with users via the user_courses table.
    - Has a **one-to-many** relationship with assignments (a course can have many assignments).

User_courses junction table

- Resolves the **many-to-many** relationship between users and courses.
- **Composite primary key:** (user_id, course_id)
- **Relationships:**
    - References users(id) and courses(id)
    - Indicates which users are enrolled in which courses.

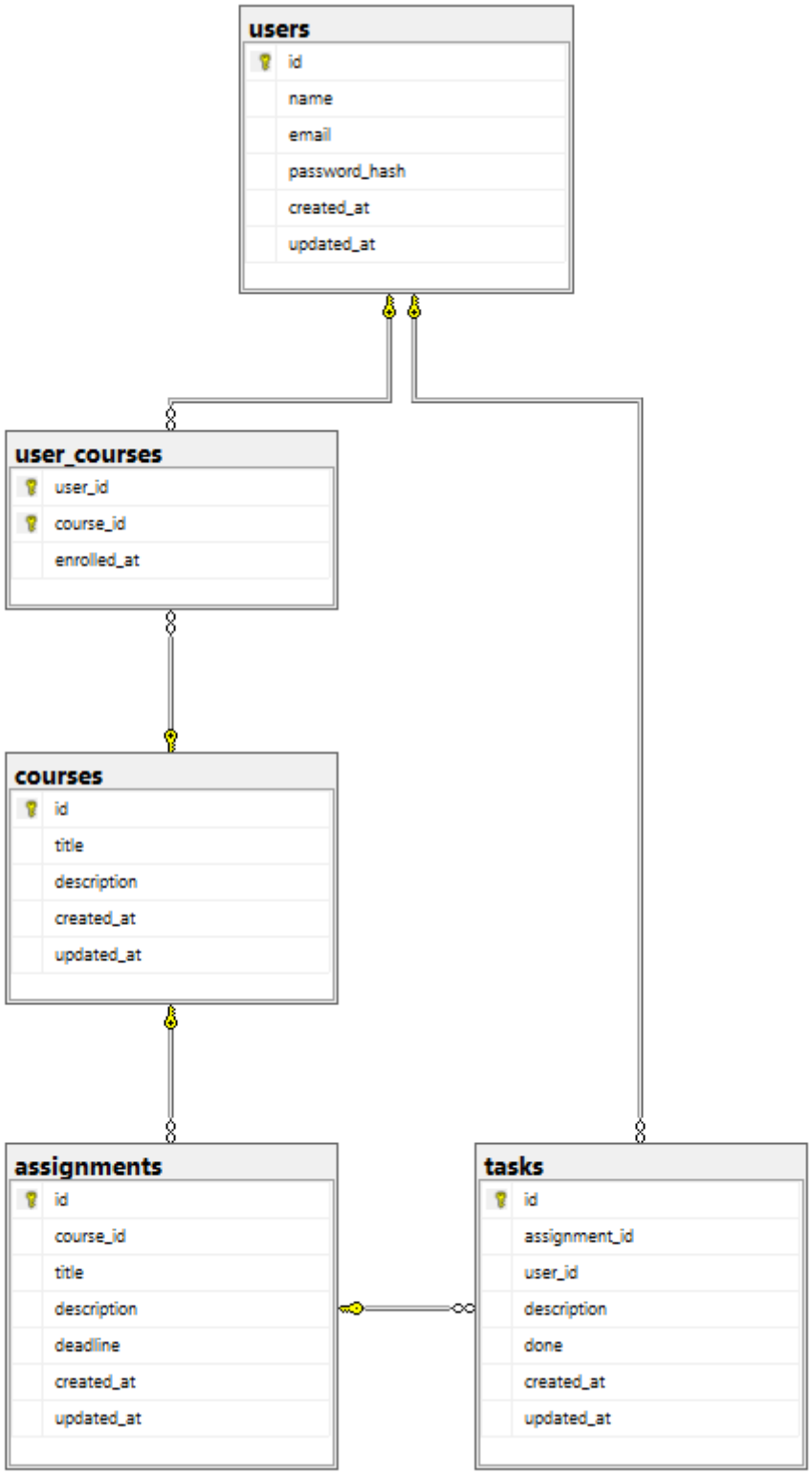
Assignments table

- Represents large tasks or projects belonging to a course.
- **Primary key:** id
- **Relationships:**
    - Belongs to one course (course_id foreign key).
    - Has a **one-to-many** relationship with tasks (each assignment can have multiple tasks, shared or personal).

Tasks table

- Represents specific actionable items linked to assignments.
- Represents a personal task created by a student for a specific assignment.
- There are no shared tasks – the lecturer only creates assignments, not tasks.
- **Primary key:** id
- **Relationships:**
    - Belongs to one assignment (assignment_id foreign key).
    - Each task must be associated with exactly one user (user_id is NOT NULL).

Diagram:


users
id
name
email
password_hash
created_at
updated_at
user_courses
user_id
course_id
enrolled_at
courses
id
title
description
created_at
updated_at
assignments
id
course_id
title
description
deadline
created_at
updated_at
tasks
id
assignment_id
user_id
description
done
created_at
updated_at