CSI 360 - Database Systems - Final Project

For your final project, you will design, develop and implement a database along with a database interface. We will be using mysql for the database and php for the interface. You are free to develop your own database idea, and I will give some guidelines below to help you flush out the scope. If you don't have ideas, we can workshop on those together. The guidelines are broken down into two pieces. The first are the requirements for the first submission and presentation. The second piece are the full requirements for your final hand-in and presentation.

1 Proposal and Presentation

Next week you will present the general idea of your database. Along with that you must include the following appropriate materials in a digital presentation. You should have two slides for the first two points and as many slides in your presentation as needed to describe your tables, according to the bullets below.

1. DB description
   (a) What is your DB? What is it designed to do?
   (b) Why are you making this database?
   (c) What could it be used for? Where could it be used?
2. ER diagram of your database idea
   (a) Must include at least 4 entities
   (b) Must include at least 3 relationships
   (c) Must include at least 1 many-to-many relationship
3. Table schema for your database idea (tables, column names and types)
   (a) Every entity has a table that fully describes it.
   (b) Relationships are appropriately included into tables.
   (c) Clearly identify primary and foreign keys.
4. FDs, Normal Forms and Uniqueness
   (a) List the Functional Dependencies in your database along with the tables that they belong with. (What can we deduce based on what? In hw3, we could tell the course name from the dept name and course number. Does something similar to this exist in your DB? Make sure to list them all.)
   (b) What is the normal form of the tables you've listed? You should push them into at least 3NF. (For example, in HW3, one of the tables was not in 3NF).
   (c) What entries in a table should be unique? (For example, in HW3, CSI 360-10 should have been unique (perhaps as a primary key?). But instead we added and used a course ID as a primary key because it uniquely identified a row in the table. Thus, the uniqueness of CSI 360-10 becomes a constraint (it also leads to a weird situation where $A \rightarrow B$ and $B \rightarrow A$). This constraint then becomes something that we must check as a constraint in our database or via our interface.)

2 Final Hand-In and Presentation

In your final hand-in you will present (in brief) updated versions of everything from the original hand in. In addition, you'll show a working website that demonstrates all aspects of your project. The following must be included in your final hand-in. As per the first presentation, everything should be presented digitally. The second part will also be a digital live demonstration.

1. Your database will end up having at least 5 tables. So, at each step in the process below, you'll want to make sure you are handling at least 5 tables. The website and demonstration should include (at a minimum) the following functionality (page names are not as important to me as the separated functionality - if you use different page names, you need to be sure):
   (a) index.php - this will be a website that links to the other launching websites. You can link to another website using html like the following:
   `<a href="make.php">refresh db</a>`
   This will include a link to make.php. On the webpage, you click on the words "refresh db", to bring you to make.php.
   (b) make.php - this should drop and then create all the tables in your database. It should include at least 4 entries in each table.
   (c) displayall.php - Display all the rows in all the tables.
   (d) data.php with associated insertn.php - You must have some way to insert new data into your database. You may follow the model of hw3 and create one data.php page if you like with a variety of insertn.phps for each different table. You may be creative and do this differently as long as you fulfill this requirement. However, your inserts (unlike hw3) should verify that no constraints are broken. You will need to do this for the relationship entries in each table. (For example, in hw3, you could insert a student-class pair in attending even though that student or class (or both) did not exist. In this project, you must check that indeed, the entry exists first. We could have accomplished this by querying the student table and ensuring that 1 row exists - same with class - before inserting into the attending table.)
   (e) A way to list information related to your many-to-many relationship. This can take the form of list/join from hw3, but I imagine you'll want to do something slightly more custom depending on your database.
   (f) One other useful page that interacts with your database. Do at least one more thing. For example, in hw3, we could have added an option to drop a student from a class.

2. In your final hand-in, you must also include updated and complete versions of items submitted for your first hand-in. This includes:
   (a) DB description
   (b) ER diagram
   (c) FDs
   (d) Normal Forms
   (e) Constraints and Uniqueness