Stage 2: Database Design

Entities:

1. **Students** represents the individual student's information needed for this system, which is modeled as an entity because students have multiple attributes and participate in relationships of all three kinds (one-to-one, many-to-one, and many-to-many). The primary key is the NetID.
2. **Instructors** represents faculty members who teach courses. This is modeled as an entity to have that information separate from courses, as there are no constraints on how many courses a Instructor can teach and vice versa. Each Instructor has a unique InstructorID as the primary key.
3. **Courses** represents the courses offered. This table has numerous attributes because the course registration system requires all this information. Each section for each semester has its tuple, and the primary key is the CRN and semester taught in combination because neither is unique on its own.
4. **Degrees** represents the degrees that students can pursue, and the table consists of their specific requirements. This is modeled as an entity to distinguish the various academic programs and their requirements. For this project, we will only focus on MCS and MS CS to ease the load, but the table could be updated as a stretch goal for more cohesion. The primary key is the degree name.
5. **Assistantships** represents roles (like TA or RA) that students may hold. This is modeled as an entity instead of an attribute of students because for most students, this column would likely be NULL, so we decided to compact that.
6. **Course_Reviews** represents anonymous reviews of courses, which is an entity because of its unique many-to-many relationship with the courses table, with a specific ID as the primary key.

Relationships and Cardinality Assumptions:

1. **Pursue (Students-Degrees)**: A student pursues a single degree but many students will pursue the same degree, so this is a many-to-one relationship.
2. **Enrollment (Students-Courses)**: Students can enroll in multiple courses and courses can have multiple students, so this is a many-to-many relationship.
3. **Teach (Instructors-Courses)**: An Instructor can teach multiple courses, but we will assume that each section of the course can be taught by at most one Instructor, so this is a many-to-one relationship.
4. **Hold (Students-Assistantships)**: A student can hold one assistantship and each tuple in that entity only has one student, so this is a one-to-one relationship.
5. **Waitlist (Students-Courses)**: Similar to enrollment, students can be on the waitlist for multiple courses and courses can have multiple students, so this is a many-to-many relationship. To keep order in the waitlist, the timestamp is an attribute.
6. **Have (Courses-Course_Reviews)**: Each course can have multiple reviews, and each review reflects multiple sections of a course in the course entity. As a result, this becomes a one-to-many relationship.

Logical Design/ Relational Schema:

Degrees (Degree_Name: VARCHAR(255) [PK], Breadth_Credits: INTEGER, Depth_Credits: INTEGER, Thesis: BOOLEAN, Minimum_Credits: INTEGER)

Courses (CRN: INTEGER [PK], Semester: VARCHAR(20) [PK], Course Code: VARCHAR(20), Course_Name: VARCHAR(255), Instructor_ID: VARCHAR(20), Credits: INTEGER, Breadth: VARCHAR, Department: VARCHAR, Max_Enrollments: INTEGER, Present_Enrollments: INTEGER, Is_Online: BOOLEAN, Time: VARCHAR(255), Location: VARCHAR(255))

Students (NetID: INTEGER [PK], Student_Name: VARCHAR(255), Is_International: BOOLEAN, Degree_Name: VARCHAR(255))

Instructors (Instructor_ID: INTEGER [PK], Prof_Name: VARCHAR(255), Web_Link: VARCHAR(255))

Course_Reviews (Review_ID: INTEGER [PK], CRN: INTEGER [FK to Courses.CRN], Semester: VARCHAR(20) [FK to Courses.Semester], Course_Review: VARCHAR(500), Course_Rating: REAL)

Enrollments (NetID: INTEGER [FK to Students.NetID], CRN: INTEGER [FK to Courses.NetID], Semester: VARCHAR(20) [FK to Courses.Semester])

Assistantships (NetID: INTEGER [FK to Students.NetID], Type: VARCHAR(2))

Waitlist (NetID: INTEGER [FK to Students.NetID], CRN: INTEGER [FK to Courses.CRN], Semester: VARCHAR(20) [FK to Courses.Semester], Timestamp: DATETIME)

Functional dependencies:

- **Degrees**
  Degree_Name → Breadth_Credits, Depth_Credits, Thesis, Minimum_Credits
  Degree_Name determines all other attributes in the Degrees relation.

- **Courses**
  (CRN, Semester) → Course Code, Course_Name, Instructor_ID, Credits, Breadth, Department, Max_Enrollments, Present_Enrollments, Is_Online, Time, Location
  (CRN, Semester) uniquely determines every other attribute in the Courses relation.

- **Students**
  NetID → Student_Name, Is_International, Degree_Name
  NetID determines all other attributes in the Students relation.

- **Instructors**
  Instructor_ID → Prof_Name, Web_Link
  Instructor_ID determines all other attributes in the Instructors relation.

- **Course_Reviews**
  Review_ID → CRN, Semester, Course_Review, Course_Rating
  Review_ID determines all other attributes in the Course_Reviews relation.

- **Enrollments**
  (NetID, CRN, Semester) → ∅
  (NetID, CRN, Semester) uniquely identifies each row, but there are no other attributes determined by this key directly in the Enrollments table. The absence of additional attributes means there are no functional

dependencies within this relation beyond the identification of each enrollment record.

- **Assistantships**
  NetID → Type
  Assuming each NetID can have at most one type of assistantship, NetID determines the Type of assistantship.

- **Waitlist**
  (NetID, CRN, Semester) → Timestamp
  (NetID, CRN, Semester) determines the Timestamp when the student was added to the waitlist.

Analyzing Each Relation for 3NF:

- **Degrees**
  1NF: Yes, as each attribute contains only atomic values.
  2NF: Yes, because it has a single field as the primary key, eliminating the possibility of partial dependencies.
  3NF: Yes, there are no transitive dependencies between non-primary key attributes.

- **Courses**
  1NF: Yes, all attributes have atomic values.
  2NF: Yes, the composite primary key is (CRN, Semester), and all non-key attributes are fully functionally dependent on the entire primary key.
  3NF: Yes, there are no transitive dependencies between non-primary key attributes.

- **Students**
  1NF: Yes, since all attributes contain atomic values.
  2NF: Yes, as it has a single primary key (NetID), thus no partial dependency can exist.
  3NF: Yes, since there are no transitive dependencies.

- **Instructors**
  1NF: Yes, all attributes are atomic.
  2NF: Yes, it has a single attribute primary key.
  3NF: Yes, there are no transitive dependencies.

- **Course_Reviews**
  1NF: Yes, attributes are atomic.
  2NF: Yes, the primary key is Review_ID, and all other attributes are fully functionally dependent on it.
  3NF: Yes, since there are no transitive dependencies.

- **Enrollments**
  1NF: Yes, as it follows the rule of atomicity.
  2NF: Yes, considering its composite key (NetID, CRN, Semester) directly relates to the enrollment record without partial dependency.
  3NF: Yes, since there are no transitive dependencies.

- **Assistantships**
  1NF: Yes, attributes are atomic.
  2NF: It's in 2NF if we assume each NetID can only have one type of assistantship.
  3NF: Yes since there are no transitive dependencies between non-key attributes.

- **Waitlist**
  1NF: Yes, given the atomicity of attributes.
  2NF: Yes, as the composite key uniquely identifies each record, and there are no partial dependencies.
  3NF: Yes, the Timestamp is directly dependent on the composite key without any transitive dependency.