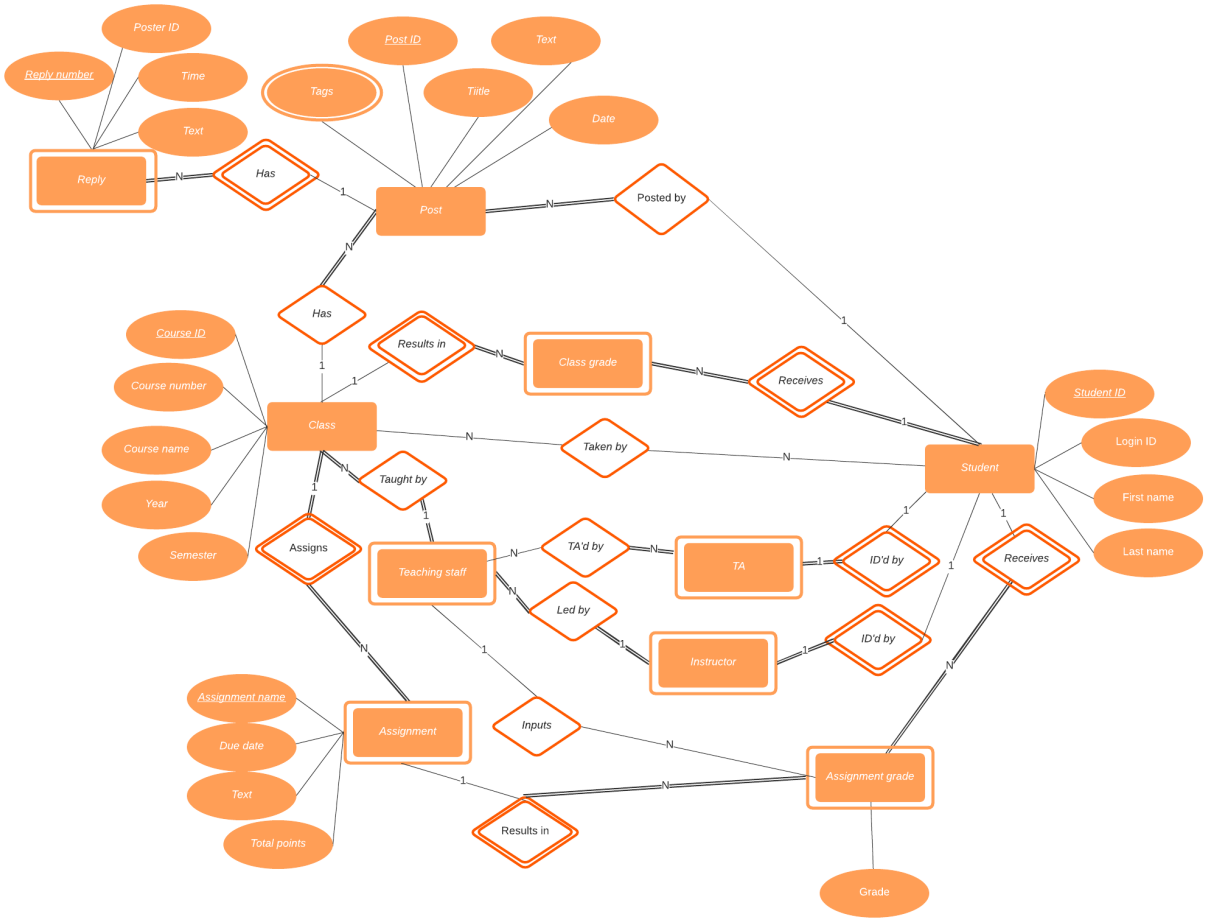
final project

Question 1: Database Design (15 points)

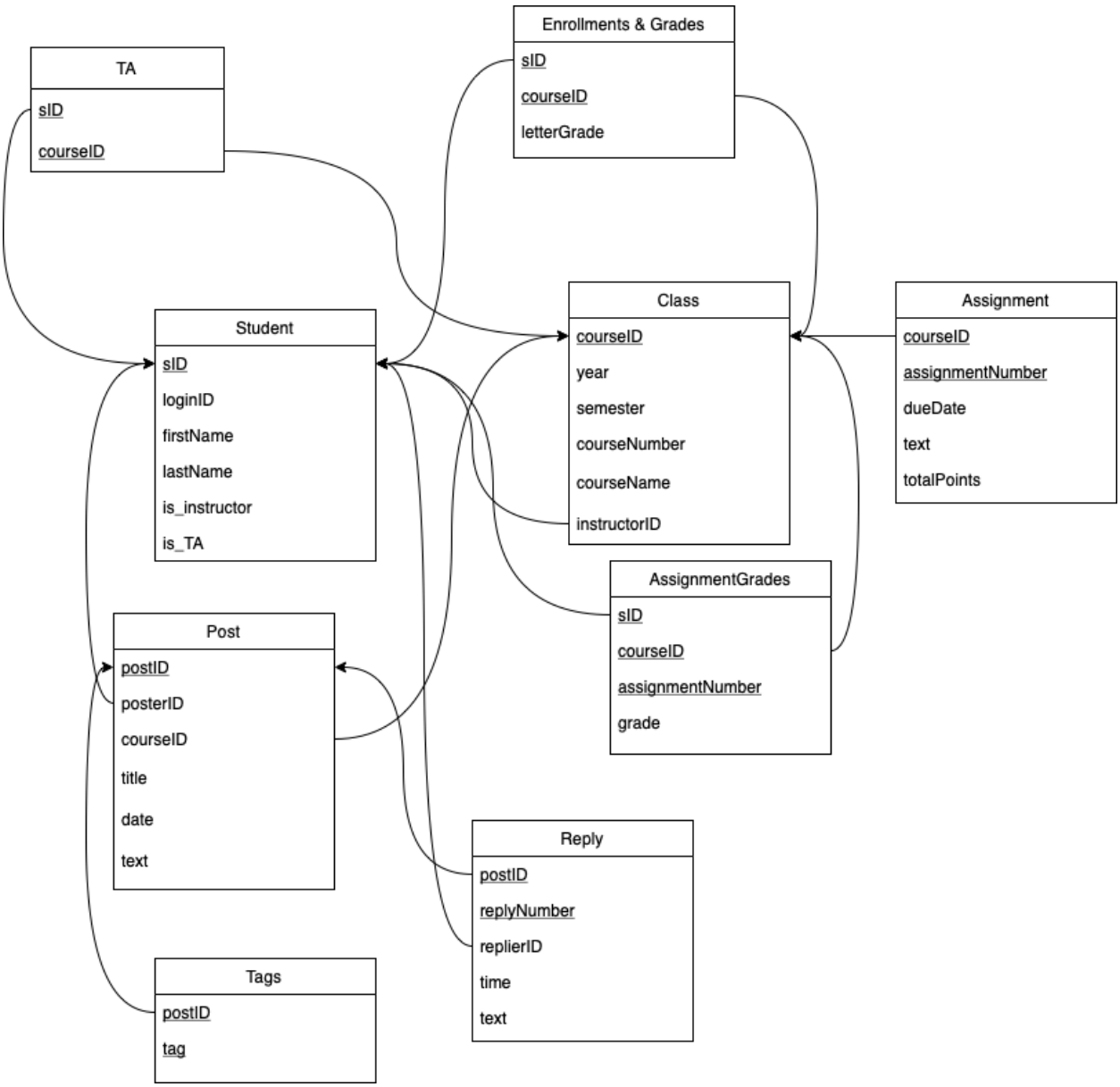


Assumptions:

- There must be one and only one instructor per class
- An instructor can teach more than one class
- A course must have at least one assignment
- Each assignment grade can only be input by one person
- Who posted a post reply is identified by their student ID
- A class may have no students

- A student can TA multiple classes
- TAs and instructors all count as students and have a unique student ID
- A student must receive at least one assignment grade for a class and may not have a final letter grade for a class (if the class is in progress)
- An assignment may or may not be graded
- All assignments are individual assignments (i.e. no group assignments)
- Each post has a unique post ID
- Each tag for a post must be unique
- Each reply in a thread is numbered
- The assignments in each class are numbered
- Every course has a unique course ID
- A letter grade is A-F with an option - or + at the end
- No two assignments within a course have the same name

Question 2: Relational Model Creation (15 points)



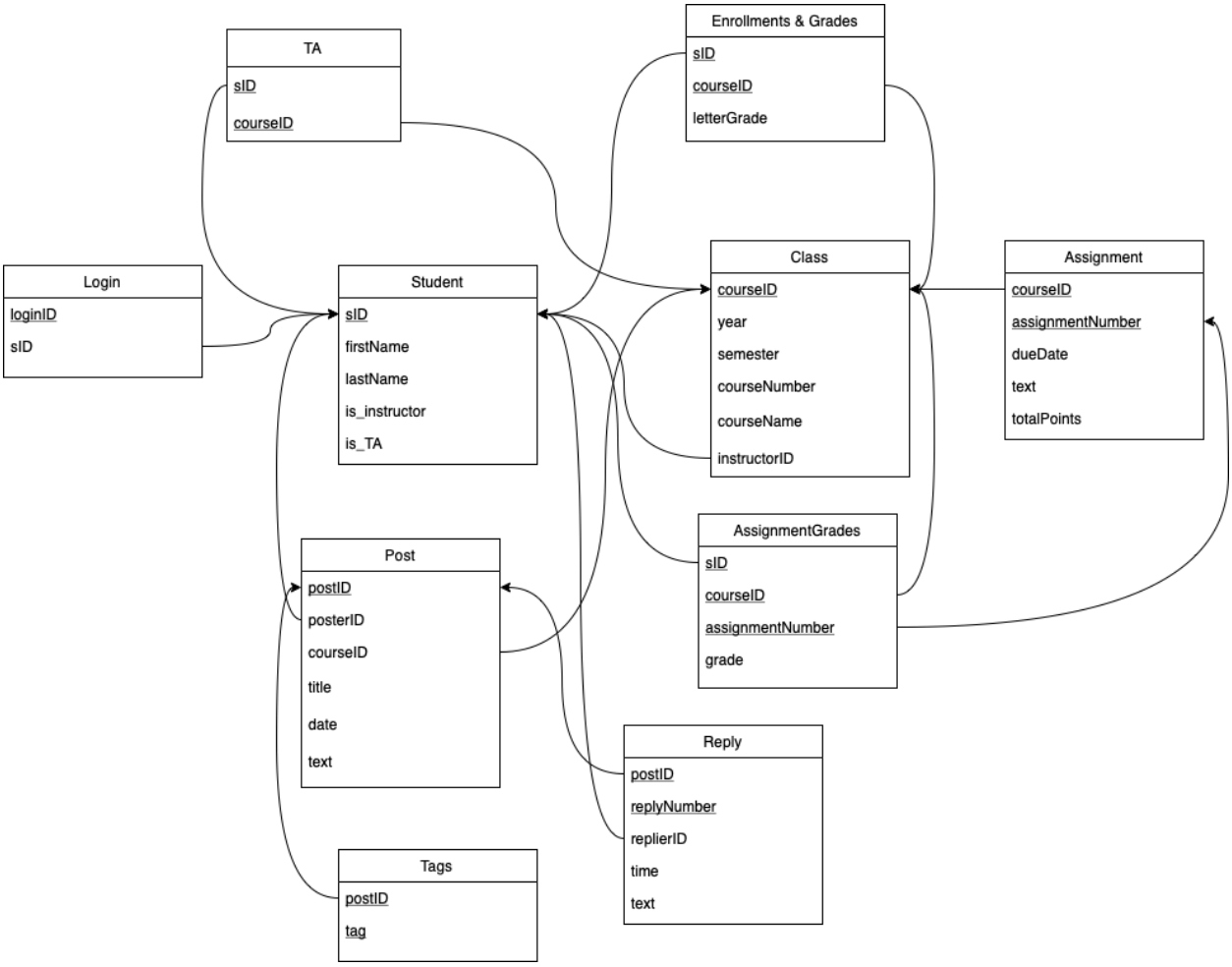
Question 3: Database Normalization (15 points)

The following are the functional dependencies that I think should hold in my application after normalization to 3NF.

- courseID → year, semester, courseNumber, courseName, instructorID
- sID → loginID, firstName, lastName, is_instructor, is_TA
- courseID, assignmentNumber → dueDate, text, totalPoints

- postID → posterID, courseID, title, date, text
- postID, replyNumber → replierID, time, text

The new relational model:



Question 4: GCP WebServer (10 points)

Hyperlink to the web application: http://35.193.140.81/index.php

Question 5: SQL Database Creation (5 points)

See `createdatabase.sql` and `dropdatabase.sql`.

Question 6: Database Population (20 points)

(a) (10 points) Take the data in the `canvas.csv` and `qa.csv` file and make new `*.csv` files such that each relation has its own data file (`<relation-name>.csv`). As an example for the company database, the `department.csv` file will have 3 rows:

See the attached `.csv` files.

(b) (3 points) Create another SQL file called `populatetables.sql` that will populate all your relational tables using the bulk import capability. Here is an example of the bulk load command in MySQL using the `department.csv` file above.

See `populatetables.sql`.

(c) (7 points) Provide a database description (or documention) for your schema in your typeset PDF. As an example for the DEPARTMENT table in the company database:

The `student` table defines all of the students in the database.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION
sID	char(10)	unique identifier for student
firstName	varchar(50)	first name of student
lastName	varchar(50)	last name of student
isInstructor	boolean	1 if student is an instructor in a course, 0 otherwise
isTA	boolean	1 if student is a TA in a course, 0 otherwise

The `class` table defines all of the classes present in the database.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION
courseID	varchar(50)	unique identifier for class, made by combining the course number with the semester and year
year	int	year the class is offered
semester	varchar(6)	semester the class is offered (fall or spring)
courseNumber	varchar(50)	course number, e.g. "CS377"
courseName	varchar(100)	name of the course, e.g. "Introduction to Computer Science II"
instructorID	char(10)	unique identifier for the course instructor

The `login` table maps all login IDs to their corresponding student identifiers.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION
loginID	varchar(50)	ID string used by student to log in
sID	char(10)	corresponding unique student identifier

The `ta` table maps the student IDs of all TA's to the class(es) they TA for.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION
sID	char(10)	unique student identifier for TA
courseID	varchar(50)	unique identifier for the course that the student TA's for

The `enrollment_grade` table shows the classes that the students have been enrolled in as well as the final grade they've received for the course.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION
sID	char(10)	unique student identifier for the student
courseID	varchar(50)	unique identifier for the course that the student took/is taking
letterGrade	varchar(2)	the letter grade received for the course

The `assignment` table defines all of the assignments that have been assigned across all of the classes.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION
courseID	varchar(50)	unique identifier for class, made by combining the course number with the semester and year
assignmentName	varchar(50)	name of the assignment
dueDate	date	due date of the assignment
assignmentText	varchar(1000)	text describing the assignment
totalPoints	int	total points possible on the assignment

The `assignment_grade` table shows the grades that students received on each assignment for the classes in which they are enrolled.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION
sID	char(10)	unique identifier for student

courseID	varchar(50)	unique identifier for class, made by combining the course number with the semester and year
assignmentName	varchar(50)	name of the assignment
grade	varchar(3)	grade student received on the assignment for the course

The `post` table defines all of the posts made across all of the courses in the database.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION
postID	varchar(50)	unique identifier for the post, made by combining the course ID with the post number (sequential)
posterID	char(10)	unique identifier for the student/instructor/TA who created the post
courseID	varchar(50)	unique identifier for the class, made by combining the course number with the semester and year
title	varchar(150)	title of the post
postDate	timestamp	date the post was created
postText	varchar(1000)	content of the post

The `tag` table shows all of the tags corresponding to each post.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION
postID	varchar(50)	unique identifier for the post, made by combining the course ID with the post number (sequential)
tag	varchar(50)	name of the associated tag

The `reply` table shows all of the replies that have been made under the posts.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION
postID	varchar(50)	unique identifier for the post, made by combining the course ID with the post number (sequential)
replyNumber	int	the number associated with the reply (sequential)
replierID	char(10)	unique identifier for the student/instructor/TA who posted the reply
replyTime	timestamp	time and date the reply was posted
replyText	varchar(1000)	content of the reply

Question 7: Student Course Page (20 points)

(a) (5 points) At least one webpage that allows the student to view a particular course content (e.g., CS 377 Fall 2021).

- `homepage.php` — displays all the courses the student was enrolled in, was the instructor for, or is the TA for at one point.

(b) (5 points) Allow the student to view their current assignment grades.

- `class_page.php` — displays the name, semester, year, instructor, assignment details, assignment grades, and final grade for a student for a particular class, as well as a link to the class's Q&A page.

(c) (5 points) Allow the student to view the details of any assignment (i.e., due date, text, and total number of points).

- `class_page.php` — displays the name, semester, year, instructor, assignment details, assignment grades, and final grade for a student for a particular class, as well as a link to the class's Q&A page.

(d) (5 points) Allow the student to view their final letter grade for the course.

- `class_page.php` — displays the name, semester, year, instructor, assignment details, assignment grades, and final grade for a student for a particular class, as well as a link to the class's Q&A page.

Question 8: Teaching Staff Page(s) (30 points)

(a) (5 points) At least one webpage that allows the instructor/teaching assistant to view a particular course content (e.g., CS 377 Fall 2021).

- `instructor_class_page.php` — displays the name, semester, year, assignment details, and student grades for a course taught by the instructor. Also displays forms that allow the instructor to create a new assignment and enter a class letter grade for a student, and a link to the class Q&A page.

(b) (8 points) Create a new assignment for the course with text (e.g., Final Project worth 200 points due 12/7 at 11:59 PM).

- `instructor_class_page.php` — displays the name, semester, year, assignment details, and student grades for a course taught by the instructor. Also displays forms that allow the instructor to create a new assignment and enter a class letter grade for a student, and a link to the class Q&A page.
- `addassignment.php` — tells the instructor whether or not an assignment was successfully added after they submit the add class form, and has a link to return to the class page.

(c) (7 points) Enter grades for each student for a specific assignment.

- `assignment_page.php` — shows all the details of the assignment and the grades received on the assignment by all the students, and has a form that allows the instructor to enter a new grade for a student for the assignment.
- `enter_grade.php` — tells the instructor whether or not a grade was successfully entered after they submit the enter grade form.

(d) (5 points) Show the current grades of all students in the course for all the assignments.

- `instructor_class_page.php` — displays the name, semester, year, assignment details, and student grades for a course taught by the instructor. Also displays forms that allow the instructor to create a new assignment and enter a class letter grade for a student, and a link to the class Q&A page.
- `assignment_page.php` — shows all the details of the assignment and the grades received on the assignment by all the students, and has a form that allows the instructor to enter a new grade for a student for the assignment.

(e) (5 points) Enter the letter grade for the student at the end of the semester.

- `instructor_class_page.php` — displays the name, semester, year, assignment details, and student grades for a course taught by the instructor. Also displays forms that allow the instructor to create a new assignment and enter a class letter grade for a student, and a link to the class Q&A page.
- `enter_class_grade.php` — tells the instructor whether or not a letter grade was successfully entered after they submit the enter class grade form.

Question 9: Q&A Page(s) (35 points)

(a) (15 points) Allow the user to view existing posts and their associated threads. Posts should be displayed in the order of their timestamp. You have the freedom to choose ascending or descending.

- `qa_page.php` — displays all the posts for a class, links to their threads, and includes a form for students/instructors to add a post. Allows users to filter the posts based on tag.
- `replies.php` — displays a post and all its replies, as well as a form for a student/instructor to write a reply.

(b) (5 points) Allow the user to filter the questions and threads based on the tag of the posts.

- `qa_page.php` — displays all the posts for a class, links to their threads, and includes a form for students/instructors to add a post. Allows users to filter the posts based on tag.
- `filter_posts.php` — displays the posts that have been filtered from the Q&A page.

(c) (8 points) Allow the user to create a new post by entering the title, one or more tags that are pre-defined in the database, and the text of their post. Note that the date should automatically be populated once they submit.

- `qa_page.php` — displays all the posts for a class, links to their threads, and includes a form for students/instructors to add a post. Allows users to filter the posts based on tag.
- `create_post.php` — tells the student/instructor whether or not their post was successfully created.

(d) (7 points) Allow the user to respond to a post by entering their reply text.

- `replies.php` — displays a post and all its replies, as well as a form for a student/instructor to write a reply.
- `create_reply.php` — tells the student/instructor whether or not their reply was successfully created.

Question 10: Home/Login Page (30 points)

(a) (10 points) A webpage to “login” to the application. Users will enter their student ID and their login ID. If the student ID / login ID pair is not valid, your application should inform the user of this. Note that authentication has been significantly simplified to make it easier for the project. You do not need to support HTTP authentication, this is simply a check to make sure that there is such a user in the system.

- `index.php` — the login page of the application wherre users enter their student ID and login ID.

(b) (10 points) Once the user has logged in, they should be shown all the courses that they can access either as a student or a faculty.

- `homepage.php` — displays all the courses the student was enrolled in, was the instructor for, or is the TA for at one point.

(c) (10 points) The user should then be able to click on a specific course that takes them to the webpages built in Questions 7-9 depending on whether they are a student or memberof the teaching staff for the course in question.

- `homepage.php` — displays all the courses the student was enrolled in, was the instructor for, or is the TA for at one point.
- `instructor_class_page.php` — displays the name, semester, year, assignment details, and student grades for a course taught by the instructor. Also displays forms that allow the instructor to create a new assignment and enter a class letter grade for a student, and a link to the class Q&A page.
- `ta_class_page.php` — displays the name, semester, year, assignment details, and student grades for a course TA's by the TA. Also displays forms that allow the TA to create a new assignment and enter a class letter grade for a student, and a link to the class Q&A page.