Apoorv Chebolu (apc5fr)
Varun Karki (vk2em)
Joseph Demarchi (jsd2ej)

Database Project: TA Office Hours

Part 1 - Project Information

1.1 - Introduction

As every college student knows finding out when the teaching assistant (TA) and professor office can be extremely frustrating. There is no standardized format for professors to display their TA's office hours. Some professors have them easy to find Google Calendar while others may not even have them listed on collab. In order to facilitate students receiving the help that they need. We are developing a web based application that allows students to view the TA and the instructor office hours of the classes that they are enrolled in. The office hours will have to be entered by the instructors or TAs through a form that will be available on the web application. The students will be able to search the class that they want to view the scheduled office hours for. Once the students find the desired result, they will have the option of saving the office hours' schedule for the semester. The students' schedule will be displayed as a text table. There will be a button to allow students to ask questions to the TA and the professors before the office hours in order for the TAs to have good answers prepared for the students when they come to office hours. We believe that having a web application with the functionality listed above will be extremely useful for students, professors, and TAs alike.

All of this will be done by setting up a database system. The interface will be a web application, but all of the information will be stored in a database. We are hoping that students will spend less time looking when office hours will be and more time going to them. The students will be able to find out both the time and the locations of the office hours.

1.2 - Requirements Document

List of functions/requirements:

1. Ability to securely log into online TA Office Hours web application
2. Ability to create a new profile (student or professor)
3. Ability to securely log out of the web application.
4. Ability to search for classes
5. Ability to add classes to "My Classes". This is a bookmarking feature
6. Ability to view office hours for TAs and instructors
7. Ability to commit to an office hour session to indicate that the student/professor will be attending.
8. Ability to have a professor view and a separate student view
    a. Students should only see student view

    b. Professors should only see professor View
9. Ability for students to submit questions to the TAs before the sessions
10. Professor View:
    a. Ability for professors to view the student questions .
    b. Ability for professors to add a class to the list of classes that students can add to their “My Classes”.
    c. Ability for professors to add a office hour session.
    d. Ability for professors to add a location to host the office hours sessions.
11. Ability to export information about the classes that are in “My Classes” in JSON format.

Use Cases:

Student: Ask question-
Student user clicks Ask a Question button and fills out form.
Professor can view the question that was asked by the Student.
Professor and Student can discuss the question during office hours.

Professor: Add Class-
Professor clicks Add Class button and fills in the appropriate details.
The class gets added to the database and appears on searches when searched by student or professor users.
Professors and students can now add this new class to their “My Classes”.

Professor: Add a Session-
Professor clicks Add a Session button on the left and fills the form with appropriate information for the class the session is made.
The session now appears on “My Classes” for students and professors have that class in their “My Classes”

Professor: Add a Location-
Professor clicks on Add a location button on the left and enters a new location name.
The location now appears as an option on the dropdown menu on the form when creating a Session for a class.

Professor and Student: Export My Classes
Student or Professor user clicks on Export My class button on the left.
The user is redirected to a new page containing JSON encoded information of the classes the user has added in their “My Classes”.

Security Issues:
In order to enforce security for our web application, at an application level we have to check for SQL injection in all user input in order to prevent unwanted access by users with malicious intent.

At a database level, we have to set user privileges in order to prevent unwanted users from altering tables/data in the database.
Additionally, we will have to hash the password that are saved in the login table that is used to authenticate users.

Special Functionality-
The special functionalities that we are using are triggers, views, Foreign Key Constraints.

Export Data-
The user is going to have the option to export information about their classes in JSON format. The user will click an export button that will redirect the user to a page containing the class information encoded in JSON.

Part 2 - The Design Process

2.1 - Explanation of design decisions

2.1.1 - Why you chose the app you did?

We chose to implement a TA Office Hours application because of our own difficulties in finding office hours for all our classes. Often times the place varies for where instructors specify when and where their office hours are. Because one instructor could specify office hours in a syllabus on the resource page in Collab and another instructor could just announce them on the first day of class, we realized that there was no one stop place to find all the available office hours for all the classes that a student is taking. From this, the idea of an online web application to aid in distributing office hours information was born.

2.1.2 - Why did you choose that language?

We chose an online web application because it would increase the accessibility of the TA office hours information. Our website is platform independent and the application can be accessed on any interface that access the Internet. This means that a wide variety of devices, including desktop and mobile devices, can be used by users, which adds to the convenience and ease-of-use of our web application.

Because it was an online web application, we chose to use HTML5 and javascript for the content and CSS for the styling to make our website look more appealing. In storing and accessing information, an InnoDB database engine was used because it allowed for the use of foreign key referential-integrity constraints. Additionally, it is a transaction safe database, which allows for the database to process queries independently while avoiding race conditions. To access our database, we used php because it was a language that we were all familiar with.

2.1.3 - Should your database be secure?

Yes our database should still be secure for many reasons. First, because we are storing private information like what classes a person takes, we shouldn't give others this access. Second,

information like first and last name and email address are also stored. We also don't want to give out this information because it is a privacy concern.

2.1.4 - How did you secure it?

We secured our database through several ways. First, we used md5 hash to store our passwords.

To login and authenticate, users type a password, which is hashed and checked if its equal with the stored hashed password. Second, we as a database administrator we wanted to make sure not to have a super user and this was done by using permissions models.For instance, people who visit the website without any account will only be allowed to create a user and login. This is our basic permission model, which was granted on 'cs4750c'. Our second permissions model is the student type, which was granted on 'cs4750a'. The student permission type allows students to ask questions, view the classes that they're enrolled in, add classes, join classes, and view the JSON data of the classes they're enrolled in. Our third permissions model is the professor type, which was granted on 'cs4750b'. The professor permission type allows professors to join an existing class, view my classes, view student questions, create a class, create sessions, create locations, and view the JSON data of the classes they're enrolled in.
Additionally, we also prevented against SQL injection by preventing users from adding special escape characters.

2.1.5 - Other interesting relevant information

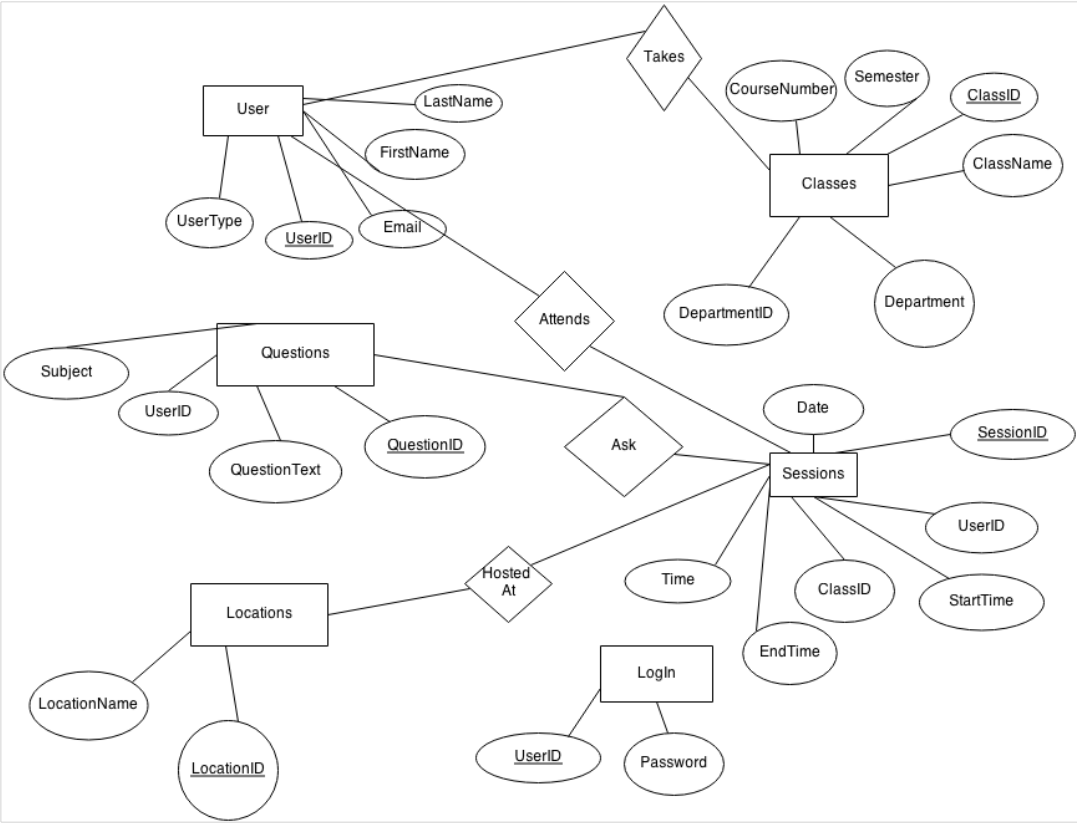
Our specialty commands include creating views, triggers, and foreign key constraints.

Our application creates two views. One view is for seeing the class view for when a user wants to see his or her classes. The second view allows for a user to join a session.

Additionally, we have a trigger for the login table. When a user is deleted through the database on the login table, the same user is also deleted in the users table.

We also use foreign keys in asks, attends, hostedAt, and takes to reference the primary keys in questions, sessions, users, locations, and classes.

2.2 - E-R Diagram



2.3 - Database schema

```
`asks` (
 `question_id` int(11) NOT NULL,
 `session_id` int(11) NOT NULL,
 KEY `question_id` (`question_id`),
 KEY `session_id` (`session_id`),
 CONSTRAINT `asks_ibfk_2` FOREIGN KEY (`session_id`) REFERENCES `sessions` (`session_id`) ON
DELETE CASCADE ON UPDATE CASCADE,
 CONSTRAINT `asks_ibfk_1` FOREIGN KEY (`question_id`) REFERENCES `questions` (`question_id`) ON
DELETE CASCADE ON UPDATE CASCADE
)

`attends` (
 `user_id` varchar(10) DEFAULT NULL,
 `session_id` int(11) DEFAULT NULL,
 UNIQUE KEY `attends_unique` (`user_id`,`session_id`),
 KEY `session_id` (`session_id`),
 CONSTRAINT `attends_ibfk_2` FOREIGN KEY (`session_id`) REFERENCES `sessions` (`session_id`) ON
DELETE CASCADE ON UPDATE CASCADE,
```

```
 CONSTRAINT `attends_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`) ON DELETE
CASCADE ON UPDATE CASCADE
)

 `classes` (
 `class_id` int(11) NOT NULL AUTO_INCREMENT,
 `department_name` varchar(50) DEFAULT NULL,
 `department_id` varchar(10) DEFAULT NULL,
 `course_number` varchar(10) DEFAULT NULL,
 `class_name` varchar(50) DEFAULT NULL,
 `semester` varchar(20) DEFAULT NULL,
 PRIMARY KEY (`class_id`),
 UNIQUE KEY `class_id` (`class_id`),
 UNIQUE KEY `class_unique` (`department_id`,`course_number`,`semester`)
)

`hostedAt` (
 `session_id` int(11) DEFAULT NULL,
 `location_id` int(11) DEFAULT NULL,
 KEY `location_id` (`location_id`),
 KEY `session_id` (`session_id`),
 CONSTRAINT `hostedAt_ibfk_2` FOREIGN KEY (`session_id`) REFERENCES `sessions` (`session_id`)
ON DELETE CASCADE ON UPDATE CASCADE,
 CONSTRAINT `hostedAt_ibfk_1` FOREIGN KEY (`location_id`) REFERENCES `locations` (`location_id`)
ON DELETE CASCADE ON UPDATE CASCADE
)

 `locations` (
 `location_id` int(11) NOT NULL AUTO_INCREMENT,
 `location_name` text,
 PRIMARY KEY (`location_id`)
)

 `login` (
 `user_id` varchar(10) NOT NULL DEFAULT '',
 `password` text,
 PRIMARY KEY (`user_id`))
`questions` (
 `question_id` int(11) NOT NULL AUTO_INCREMENT,
 `user_id` varchar(10) NOT NULL,
 `subject` text NOT NULL,
 `question_text` text NOT NULL,
 PRIMARY KEY (`question_id`)
)


 `sessions` (
 `session_id` int(11) NOT NULL AUTO_INCREMENT,
```

```
 `user_id` varchar(10) DEFAULT NULL,
 `class_id` int(11) DEFAULT NULL,
 `start_time` text,
 `end_time` text,
 `date` text,
 PRIMARY KEY (`session_id`),
 KEY `sessions_user` (`user_id`),
 KEY `sessions_class` (`class_id`),
 CONSTRAINT `sessions_class` FOREIGN KEY (`class_id`) REFERENCES `classes` (`class_id`) ON
DELETE CASCADE ON UPDATE CASCADE,
 CONSTRAINT `sessions_user` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`) ON DELETE
CASCADE ON UPDATE CASCADE
)

`takes` (
 `user_id` varchar(10) DEFAULT NULL,
 `class_id` int(11) DEFAULT NULL,
 UNIQUE KEY `takes_unique` (`user_id`,`class_id`),
 KEY `class_id` (`class_id`),
 CONSTRAINT `takes_ibfk_2` FOREIGN KEY (`class_id`) REFERENCES `classes` (`class_id`) ON
DELETE CASCADE ON UPDATE CASCADE,
 CONSTRAINT `takes_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`) ON DELETE
CASCADE ON UPDATE CASCADE
)

`users` (
 `user_id` varchar(10) NOT NULL,
 `user_type` varchar(10) DEFAULT NULL,
 `name_last` varchar(20) DEFAULT NULL,
 `name_first` varchar(20) DEFAULT NULL,
 `email` varchar(50) DEFAULT NULL,
 PRIMARY KEY (`user_id`)
)
```

2.4 - Proof that database is in 3NF

asks - 3NF because it is only two columns
attends - 3NF because it is only two columns
hostedAt - 3NF because it is only two columns
location - 3NF because it is only two columns
login - 3NF because it is only two columns
takes - 3NF because it is only two columns

As far as tables **classes**, **questions**, **sessions**, and **users** tables are concerned, as Database administrators, we kept appropriate number of attributes/columns in order to accommodate for certain JOINs that we performed during the implementation of the application.

Part 3 - Evaluation of Product

3.1 - Description of testing procedures for both database and application

In order to test the database we ran queries in MyPHP. We made sure that all of our constraints and special functions actually worked and that the queries did what we thought they did. We test the web page part by going through and making sure the all of the buttons linked to the correct page and all of the queries that were implemented in php sent or received the correct information from the database. We also dropped all of fake data and implemented new data only using the web page. By doing this we made sure that all of the functionality worked as advertised. We tested the security by make sure that a students or professors could not drop tables or have any sql statement that it does not have any person to make. We also tested the fact that if you are logged in as a student that you could not type in the teacher home page url.

3.2 - Sample data and sample queries from application

3.2.1 - Sample query 1: Shows the class information for what a particular student takes

INPUT:

SELECT user_id, classes.class_id, department_name, department_id, course_number, class_name, semester FROM classes INNER JOIN takes ON takes.class_id=classes.class_id
WHERE user_id='apc5fr'

OUTPUT:

user_id	class_id	department_name	department_id	course_number	class_name	semester
apc5fr	15	Computer Science	CS	3330	Computer Architecture	Fall 2013
apc5fr	16	Computer Science	CS	4750	Database Systems	Fall 2013
apc5fr	17	Computer Science	CS	4720	Web & Mobile Systems	Fall 2013

apc5fr	18	Applied Mathematics	APMA	2130	Differential Equations	Fall 2013
apc5fr	24	Applied Mathematics	APMA	2120	Multivar	Fall 2014

3.2.2 - Sample query 2: Shows the available office hours sessions for the particular classes that a student is taking

INPUT:

SELECT sessions.user_id, myClassView.class_id, session_id, start_time, end_time, date FROM myClassView INNER JOIN sessions ON myClassView.class_id=sessions.class_id ORDER BY class_id asc

OUTPUT:

user_id	**class_id**	**session_id**	**start_time**	**end_time**	**date**
profTest	16	53	8:00pm	9:00pm	T

3.2.3 - Sample query 3: Shows a particular office hours session for a particular user

INPUT:

*SELECT * FROM attends WHERE user_id='apc5fr' AND session_id=52*

OUTPUT:

user_id	**session_id**
apc5fr	52