Office Hour Manager: Final Report

Jonah Kim, Tammy Ngo, Yuxi Yang

Table of Contents:

DESCRIPTION:

This project was designed to help instructors—primarily in CS 3140—organize and store office hour question information. In the previous semester (Spring 2023), the instructors had used Google forms to manage the questions; however, this method lacked the necessary structure to allow for easy data analyses. This primarily stemmed from the fact that the instructors had to manually fill in all form fields. This included the student's name which caused numerous insertion anomalies where a student's name might be spelled differently across entries. Sometimes the instructor was not able to note the student's name at all which makes it hard to relate a question back to a student. This project will help address the lack of structured data in the question forms to improve the data quality.

CONNECTION:

One of our members, Jonah, is a teaching assistant for CS 3140. He believes that this app could improve and automate much of the data collection process for office hour questions. For instance, the app can automatically fill in key information such as a student's name and computing ID upon the submission of a question. It can also relate assignments to class topics. Such data can later be used to see where students are struggling in a course. If many questions relate to the same course topic, it may be a sign that there is a common misconception that most students are making that can be covered in lecture. Additionally, by keeping track of office hours' and questions' times, instructors can see when the most questions are being asked and can arrange for more office hours to be held at those times.

OVERVIEW and REQUIREMENTS:

The system should allow students to log onto the website with their computing id and view active office hours for courses which they are enrolled in. Students can submit office hour questions which will appear in the Instructor views where they can be answered. The questions should be either tied to an assignment or a list of topics. These questions can be edited/commented on before finally being closed.

Privileged users should be able to enroll other users into courses as Students or Instructors. The privileged users also should be able to create Courses, Assignments, and other necessary information to set up class office hours. All instructors can add a new Location to the database where they can host office hours. The different user views should be protected by authentication permissions. That is, a Student should not be able to navigate to a Professor view, and vice versa.

Core User Stories:

Accounts:

- A User can signup and select their relevant account options.
- A UniversityMember can sign onto the app using their computing id and be redirected to their relevant home page.
- A User should not be able to access other account restricted pages without the necessary permissions.

Instructors:

- An Instructor can create an OfficeHourSession by selecting the duration, Location, and Class.
  - An Instructor can add a Location to the database so they do not have to enter one manually each time.
- An Instructor can view their active OfficeHourSessions and see the questions submitted to them.
  - An Instructor should be able to add private instructor comments or add extra topics to any question.
  - An Instructor should be able to close a question to mark it as finished.
  - An Instructor should be able to close an OfficeHour session to mark it as finished.
- An Instructor can view all OfficeHourQuestions and OfficeHourSessions for their Class(es).

Students:

- A Student should be able to see any OfficeHourSession that are currently available for the classes they are enrolled in.
- A Student can submit an OfficeHourQuestion to an OfficeHourSession.
  - A Student can edit any *open* OfficeHourQuestion that they submitted.
  - A Student can withdraw an *open* OfficeHourQuestion.
  - A Student should be able to supply extra comments on their OfficeHourQuestion in addition to linking it to an Assignment or CourseTopic.

Professor:

- A Professor can create a Course.
- A Professor can create a Class (i.e., a Course with a Semester linked).
- A Professor can create a list of CourseTopics taught in a Course.
- A Professor can create a list of Assignments taught in a Course, and link CourseTopics to that Assignment.

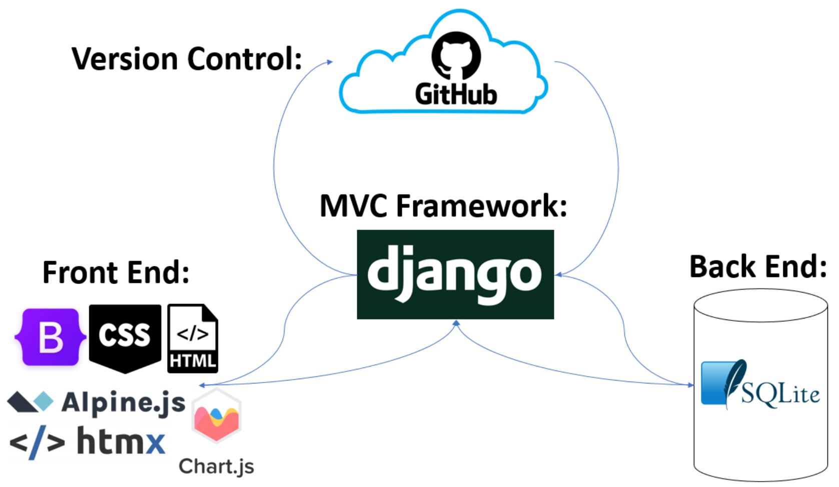
RESPONSIBILITIES:

- **Jonah**:
    - Stakeholder and Designer: Lists requirements and informs design
    - Back-End: Python, Django, SQLite
    - Front-End: HTML, CSS, Javascript, Alpine.js, HTMX, Chart.js
- **Tammy**:
    - Back-End: Python, Django, SQLite
    - Front-End: HTML, Javascript, Chart.js
- **Yuxi**:
    - Back-End: Python, Django, SQLite
    - Front-End: HTML, CSS, Javascript, Alpine.js, HTMX

ARCHITECTURE:

Figure 1

System Architecture



This project will employ a three-tiered architecture using the Django MVT/MVC framework. Our frontend will primarily use HTML, CSS, and Bootstrap5 to render. To encode advanced controls and functionality, we will employ the javascript frameworks Alpine.js, HTMX, and Chart.js. Alpine.js will be used to control our forms; HTMX will be used to initiate asynchronous requests and HTML replacement; and Chart.js will be used for information summaries. The controller will be implemented in Python using Django. The backend/model will be connected to an SQLite3 database. We will use GitHub as our version control system to help facilitate group contributions. A diagram (**Figure 1**) of the key information is provided above.

BUSINESS RULES:

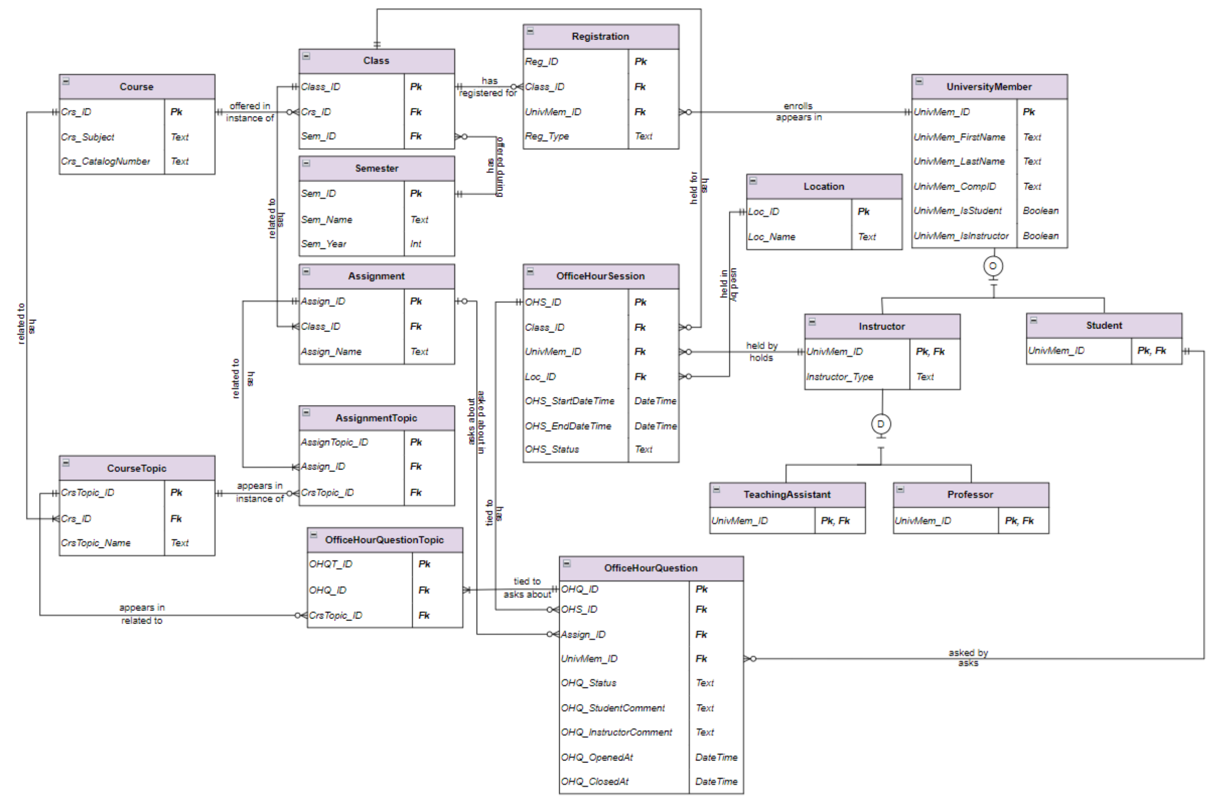
- Accounts:
  - A TA is a type of Instructor
  - A TA is a type of UniversityMember
  - A Professor is a type of Instructor
  - A Professor is a type of UniversityMember
  - A Student is a type of UniversityMember
  - A UniversityMember must be either a Student or (inclusive) Instructor.
  - An Instructor must be either a TA or (exclusive) a Professor.
- Instructor:
  - An Instructor can optionally hold one or more OfficeHourSessions
  - An Instructor can optionally teach/assist in one or more Courses/Classes
  - An Instructor is required to be either a TA or (exclusive) a Professor for a Class.
- Student:
  - A Student can optionally take one or more Courses/Classes
  - A Student can optionally make one or more OfficeHourQuestions
- OfficeHourSession:
  - An OfficeHourSession is mandatorily associated with exactly one Instructor
  - An OfficeHourSession is mandatorily associated with only one Course/Class
  - An OfficeHourSession can optionally have zero or more OfficeHourQuestions
  - An OfficeHourSession is mandatorily associated with only one Location
- OfficeHourQuestion:
  - An OfficeHourQuestion can optionally be associated with one Assignment
  - An OfficeHourQuestion can optionally be associated with one or more CourseTopics
  - An OfficeHourQuestion must be either associated with an Assignment or (inclusive) with one or more CourseTopics
  - An OfficeHourQuestion is mandatorily associated with exactly one OfficeHourSession
  - An OfficeHourQuestion is mandatorily associated with exactly one Student
- Course:
  - A Course can have zero or many Students
  - A Course can have one or many Instructors
  - A Course can have zero or many OfficeHourSession
  - A Course can have one or many Assignments
  - A Course can have one or many CourseTopics
  - A Course/Class must be associated with exactly one Semester
- Course Topic:
  - A CourseTopic can optionally be associated with one or more Assignments
  - A CourseTopic can optionally be associated with one or more OfficeHourQuestions
  - A CourseTopic is mandatorily associated with exactly one Course
- Semester:
  - A Semester may appear in many Courses

- Assignment:
    - An Assignment can have one or more CourseTopics
    - An Assignment is mandatorily associated with exactly one Course/Class
    - An Assignment can optionally be asked about in one or more OfficeHourQuestions.
- Location:
    - A Location can optionally be associated with one or more OfficeHourSession
- Actions:
    - Any UniversityMember can view OfficeHourSession for classes that they are enrolled in as a Student
    - An Instructor can create OfficeHourSession
    - An Instructor can create Locations
    - A Student can create OfficeHourQuestions
    - OfficeHourQuestions can be edited and reviewed by the Instructor who hosted the OfficeHourSession where they were submitted.
    - A Professor can create/edit Courses
    - A Professor can create/edit Assignments
    - A Professor can create/edit CourseTopics
- Constraints:
    - An Instructor can hold OfficeHourSessions only for Classes in which they are enrolled as a TA or Professor.
    - A Student can submit OfficeHourQuestions only to OfficeHourSessions which are being held for a Class that they are enrolled in as a Student.
    - A TA can access both TA views and Instructor views.
    - A Professor can access both Professor views and Instructor views.
    - A Student can only access Student views.
        - (**Note**: There are overlapping subtypes so a UniversityMember may both a Student and a TA)

ENTITY RELATIONSHIP DIAGRAM:

Figure 2

Entity Relationship Diagram (ERD)



VIEWS:

Account/Common:

Figure 3

Sign-Up

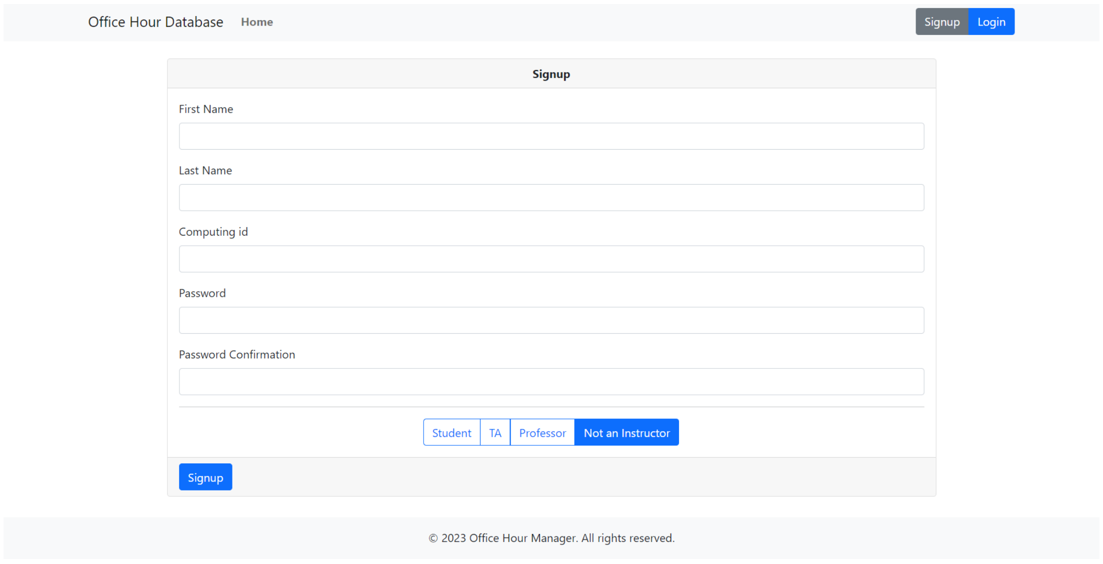


Figure 4

Login

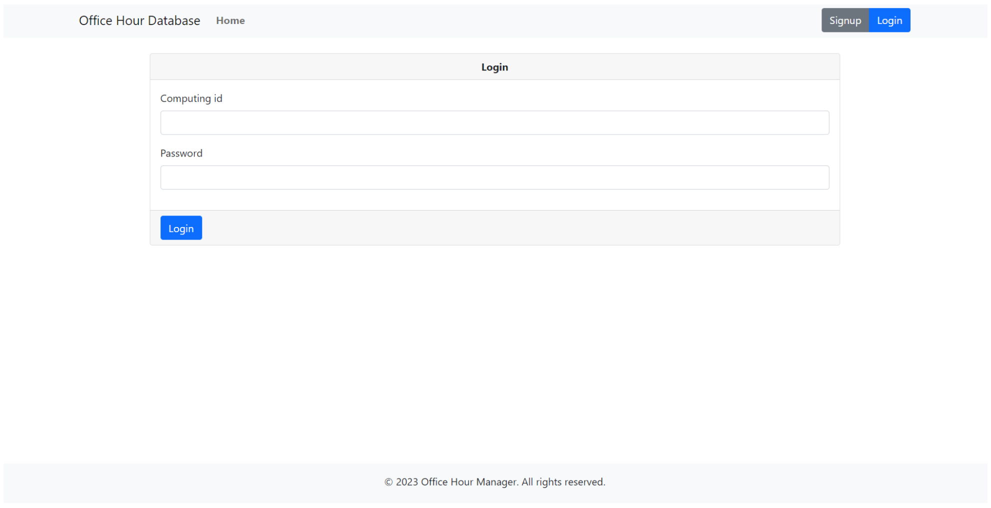
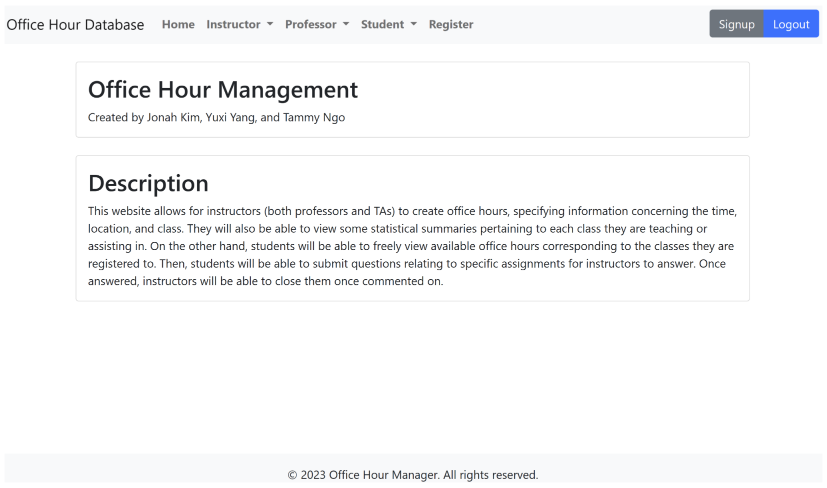


Figure 5

Home Screen



Office Hour Management

Created by Jonah Kim, Yuxi Yang, and Tammy Ngo

Description

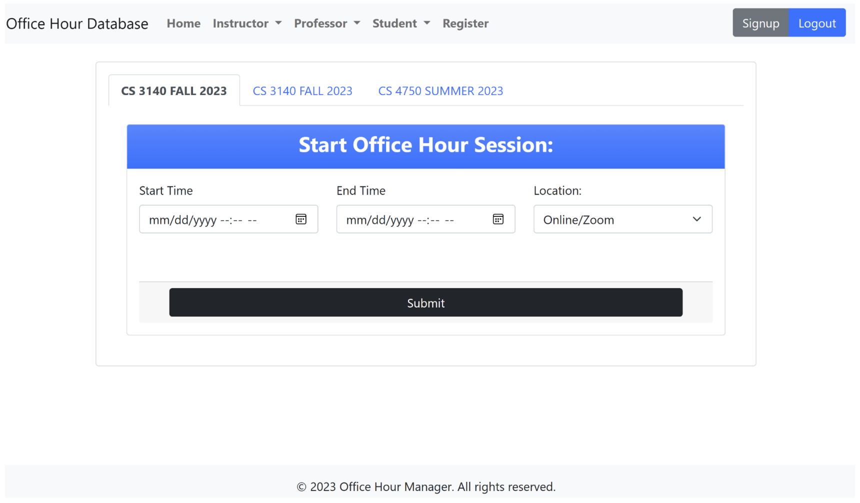
This website allows for instructors (both professors and TAs) to create office hours, specifying information concerning the time, location, and class. They will also be able to view some statistical summaries pertaining to each class they are teaching or assisting in. On the other hand, students will be able to freely view available office hours corresponding to the classes they are registered to. Then, students will be able to submit questions relating to specific assignments for instructors to answer. Once answered, instructors will be able to close them once commented on.

© 2023 Office Hour Manager. All rights reserved.

Instructor:

Figure 6

Start Office Hour Session



Start Time: mm/dd/yyyy --:-- --

End Time: mm/dd/yyyy --:-- --

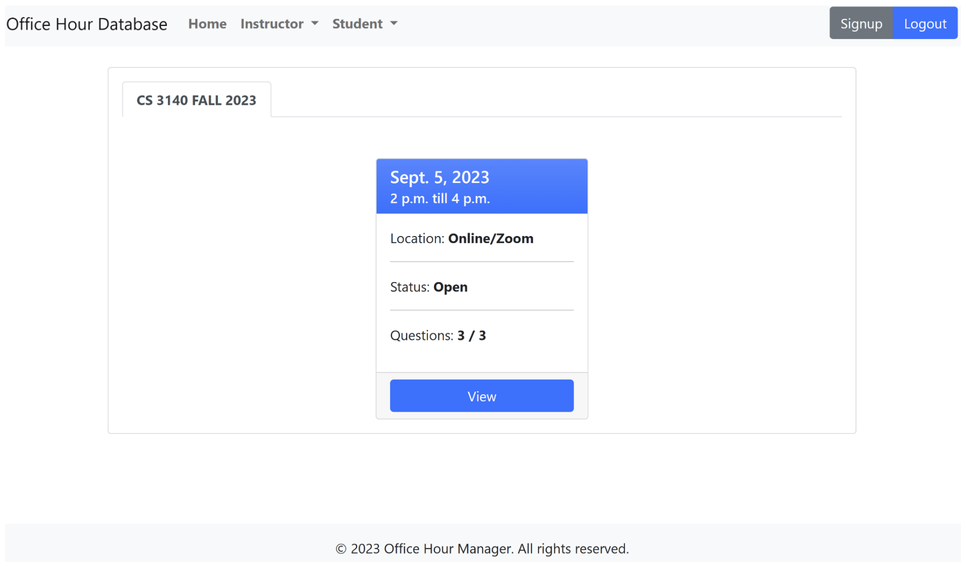
Location: Online/Zoom

© 2023 Office Hour Manager. All rights reserved.

Note: Originally, there was a different screen for creating a location, but now locations can be created from the dropdown menu with a modal popup.

Figure 7

Your Office Hours: List



Sept. 5, 2023
2 p.m. till 4 p.m.

Location: **Online/Zoom**

Status: **Open**

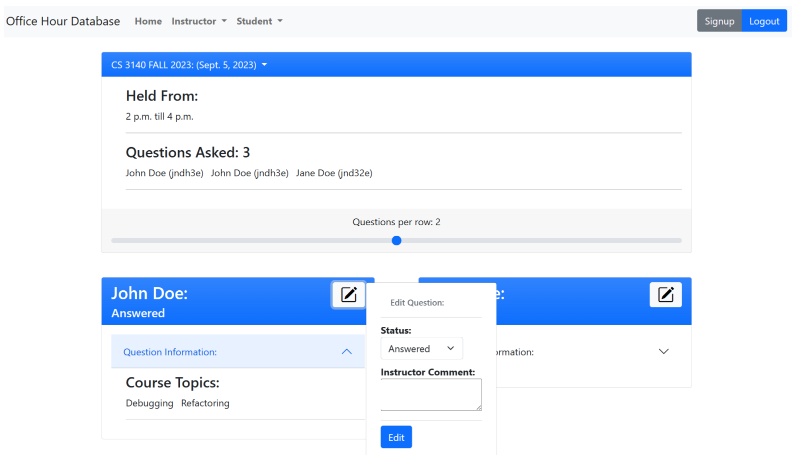
Questions: **3 / 3**

© 2023 Office Hour Manager. All rights reserved.

Note: This view lists an individual instructor's past and current office hours. As seen above, it includes information about the office hour's location, status, and pending questions.

Figure 8

Your Office Hours: Detail



Held From:
2 p.m. till 4 p.m.

Questions Asked: 3
John Doe (jndh3e) John Doe (jndh3e) Jane Doe (jnd32e)

Questions per row: 2

John Doe:
Answered

Course Topics:
Debugging Refactoring

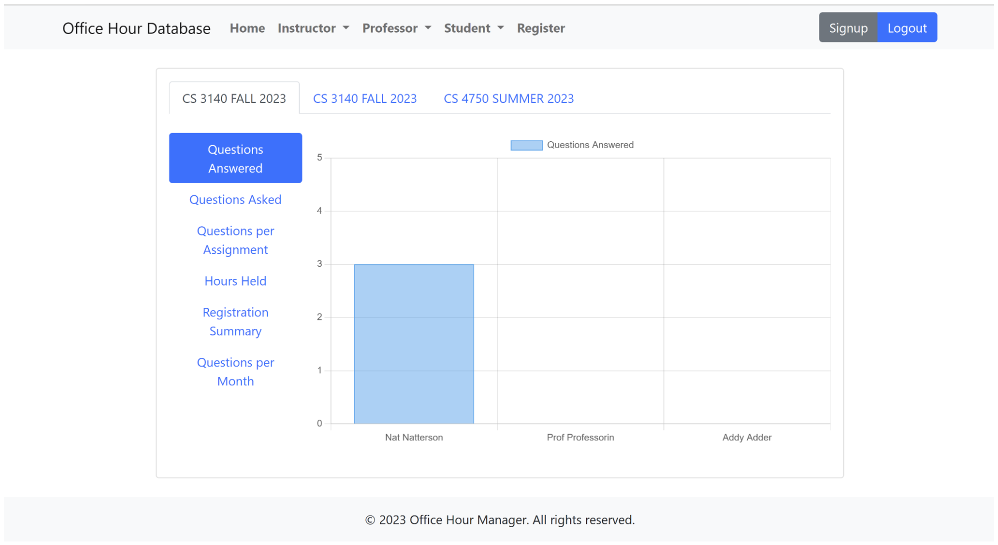
Status: Answered

Instructor Comment:

Note: Accessed through clicking *View* on **Figure 7**. One can view questions asked in the specified office hours, can change how many questions can be seen on each row, and can edit the status and reply to the student question.

Figure 9

Information Summaries



Note: For future development, there will be more screens and visualizations displayed here. The graphs here are the starting point.

Professor:

Figure 10

Create Course

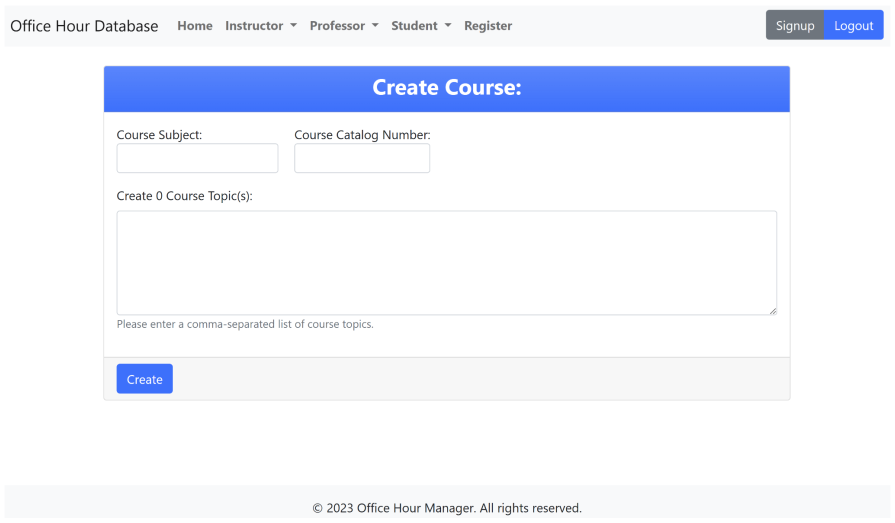
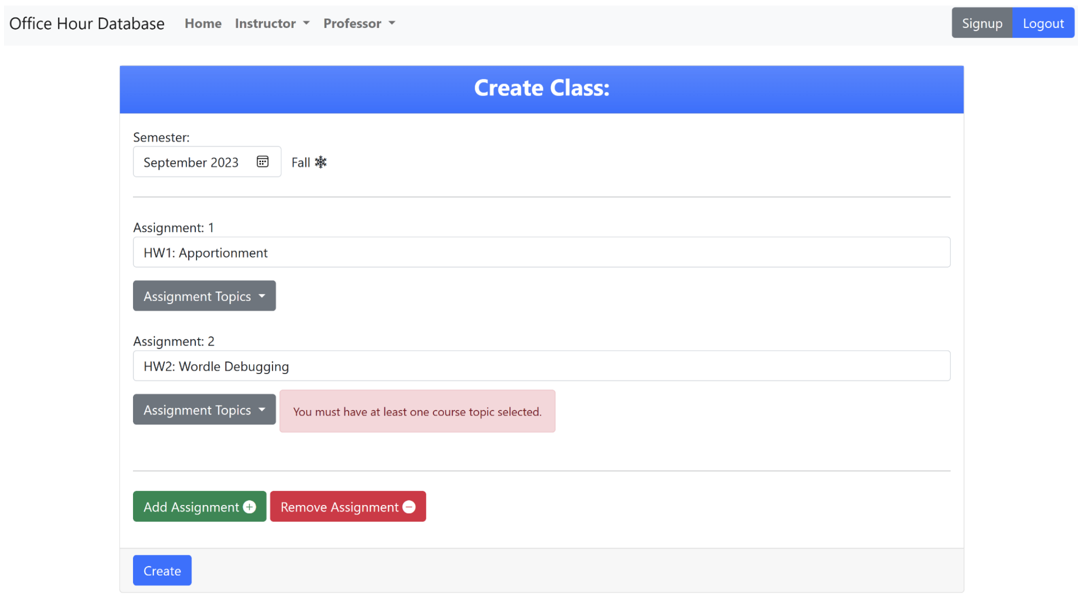


Figure 11

Create Class



Create Class:

Semester:
September 2023 Fall ❄

Assignment: 1
HW1: Apportionment
Assignment Topics

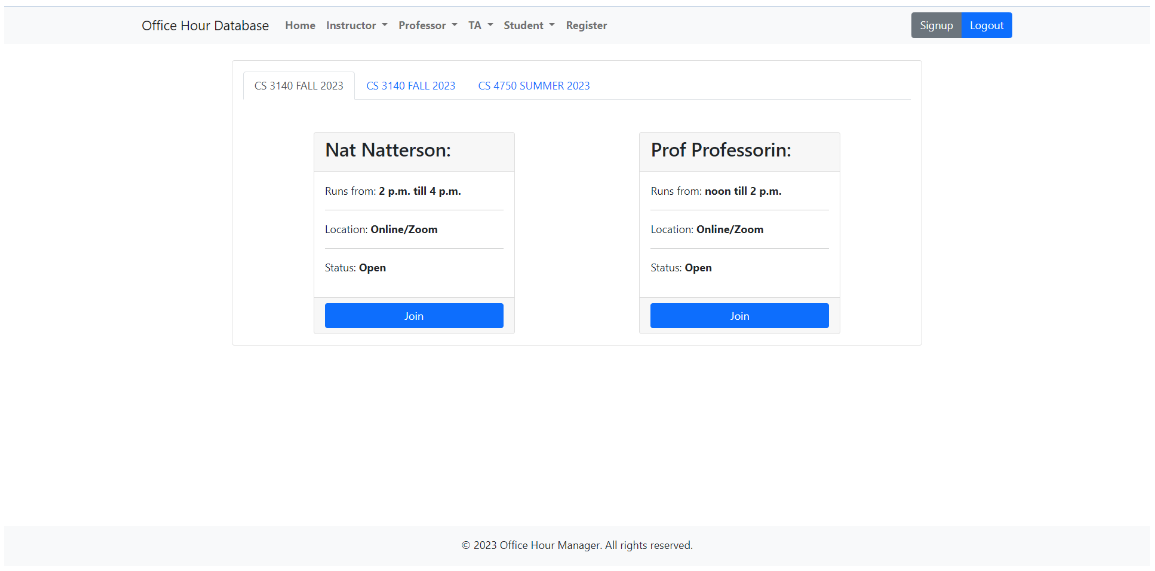
Assignment: 2
HW2: Wordle Debugging
Assignment Topics
You must have at least one course topic selected.

Add Assignment
Remove Assignment
Create

Student:

Figure 12

Available Office Hours



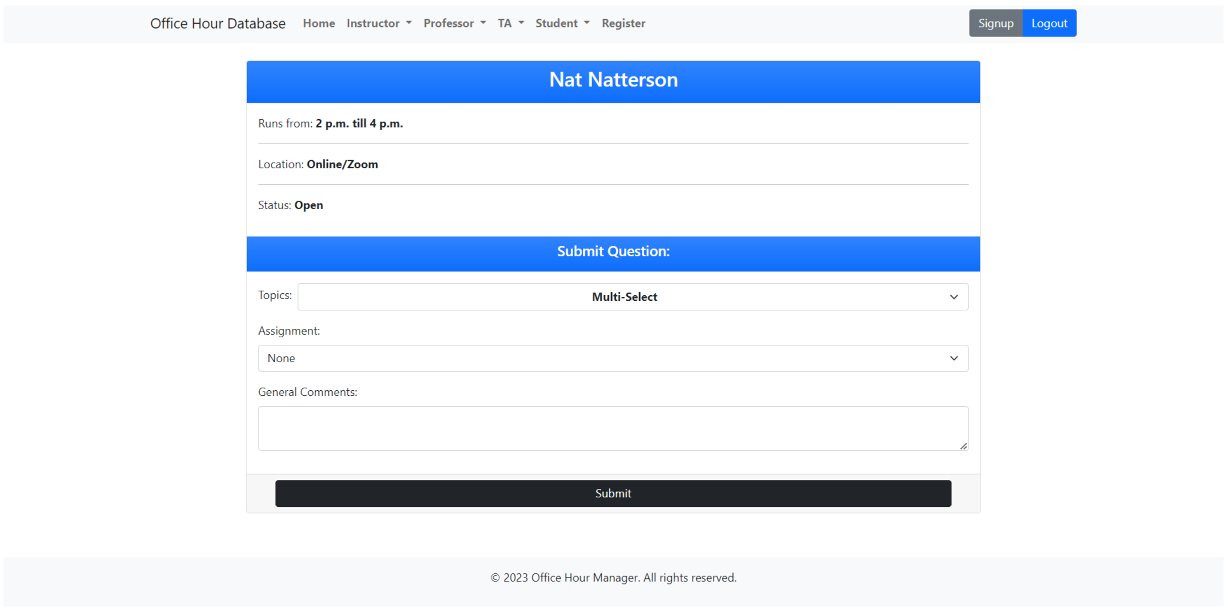
Nat Natterson:
Runs from: **2 p.m. till 4 p.m.**
Location: **Online/Zoom**
Status: **Open**
Join

Prof Professorin:
Runs from: **noon till 2 p.m.**
Location: **Online/Zoom**
Status: **Open**
Join

© 2023 Office Hour Manager. All rights reserved.

Figure 13

Create Question



Nat Natterson

Runs from: **2 p.m. till 4 p.m.**

Location: **Online/Zoom**

Status: **Open**

Submit Question:

Topics: **Multi-Select**

Assignment: None

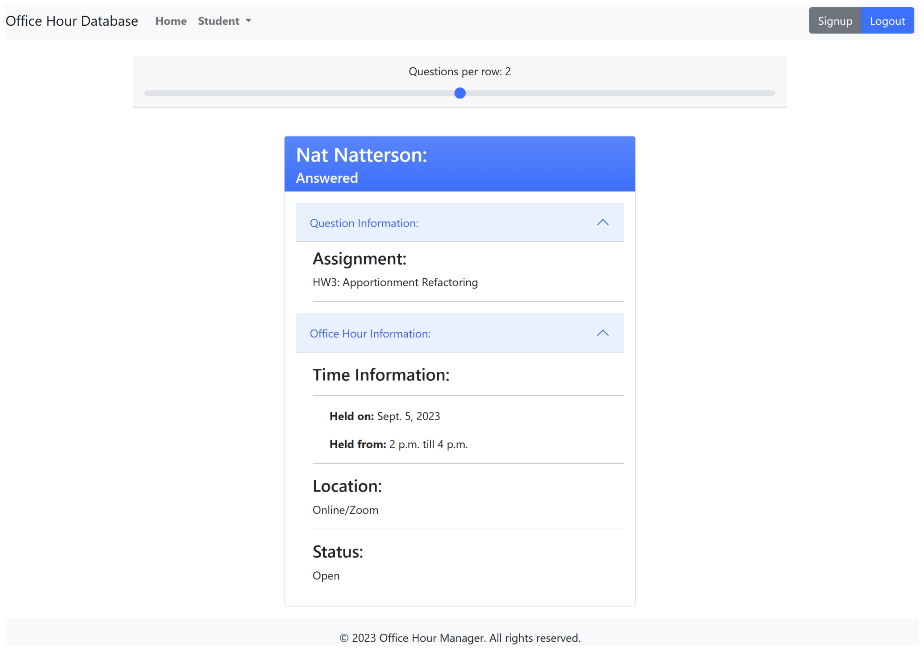
General Comments:

Submit

© 2023 Office Hour Manager. All rights reserved.

Figure 14

View Question



Questions per row: 2

Nat Natterson:
Answered

Question Information:

Assignment:
HW3: Apportionment Refactoring

Office Hour Information:

Time Information:

Held on: Sept. 5, 2023

Held from: 2 p.m. till 4 p.m.

Location:
Online/Zoom

Status:
Open

© 2023 Office Hour Manager. All rights reserved.

POSSIBLE IMPROVEMENTS:

There are numerous possible improvements that can be made to the project. What follows are features that were planned or desired but were ultimately deemed of secondary importance to the core actions. We list them here for future development.

feat. 1: A UniversityMember can sign onto the app using their computing id and be redirected to their relevant home page. (*We prioritized other functionality during development though this should be relatively simple to implement in the future.*)

feat. 2: An Instructor should be able to close an OfficeHour session to mark it as finished. (*Students are only able to see office hours with valid times, and instructors usually do not finish an OHS early so this feature was deemed unimportant*.)

feat. 3: A Student can edit any *open* OfficeHourQuestion that they submitted. (*From experience, most students do not put comments on their questions; therefore, this feature was not prioritized during development.*)

feat. 4: An instructor can be a TA for multiple offerings of the same Class during the same Semester. (*Currently, our app does not smartly detect when this occurs; therefore, one might see two Classes that look identical but are actually different sections.*)

feat. 5: A Professor or superuser can enroll UniversityMembers into a Class by supplying a list of computing ids. (*Since enrollment is an action that will only be performed once or twice during a Semester, we prioritized other user stories; however, a Registration page is available to Superusers where they can manually register UniversityMembers for Classes.* ***Note****, this means that new accounts must be enrolled manually when testing.*)

feat. 6: The website should update the list of OHQs for an Instructor in the background without needing to refresh the page manually *(requires HTMX to implement*).

feat. 7: An Instructor can filter through their past OHQs. (*We provide simple filters in our web app but nothing advanced.*)

feat. 8: OfficeHourSessions should allow multiple Instructors.

feat. 9: OfficeHourSessions should be created from an OfficeHour entity which stores primarily date information. This would allow an Instructor to create OfficeHourSessions from an OfficeHour without having to manually specify a start and end time manually every time.

feat. 10: An Office hour session should have a temporary Queue object (or equivalent functionality) to help smartly filter Students based on their priority. (*Our OHQs are sorted by status and then by time which was deemed sufficient for the project's scope.*)

SQL QUERIES:

CREATE TABLE QUERIES:

UniversityMember:

```
CREATE TABLE UniversityMember (
   UnivMem_ID INTEGER NOT NULL
      PRIMARY KEY AUTOINCREMENT,
   UnivMem_FirstName TEXT NOT NULL,
   UnivMem_LastName TEXT NOT NULL,
   UnivMem_CompID TEXT UNIQUE NOT NULL
      CHECK (length(UnivMem_CompID) <= 8),
   UnivMem_IsStudent BOOLEAN NOT NULL,
   UnivMem_IsInstructor BOOLEAN NOT NULL
);
```

Student:

```
CREATE TABLE Student (
   UnivMem_ID INTEGER NOT NULL
      PRIMARY KEY AUTOINCREMENT
      REFERENCES UniversityMember(UnivMem_ID) ON DELETE CASCADE
);
```

Instructor:

```
CREATE TABLE Instructor (
   UnivMem_ID INTEGER NOT NULL
      PRIMARY KEY AUTOINCREMENT
      REFERENCES UniversityMember(UnivMem_ID) ON DELETE CASCADE,
   Instructor_Type TEXT NOT NULL
      CHECK (Instructor_Type in ('TA', 'PROF'))
);
```

Professor:

```
CREATE TABLE Professor (
   UnivMem_ID INTEGER NOT NULL
      PRIMARY KEY AUTOINCREMENT
      REFERENCES Instructor(UnivMem_ID) ON DELETE CASCADE
);
```

TeachingAssistant:

```
CREATE TABLE TeachingAssistant (
   UnivMem_ID INTEGER NOT NULL
```

```
        PRIMARY KEY AUTOINCREMENT
        REFERENCES Instructor(UnivMem_ID) ON DELETE CASCADE
);
```

Semester:

```
CREATE TABLE Semester (
    Sem_ID INTEGER NOT NULL
        PRIMARY KEY AUTOINCREMENT,
    Sem_Name TEXT NOT NULL
            CHECK (Sem_Name IN ('SPRING', 'SUMMER', 'FALL')),
    Sem_Year DATE NOT NULL
            CHECK (Sem_Year >= 1900)
);
```

Location:

```
CREATE TABLE Location (
    Loc_ID INTEGER NOT NULL
        PRIMARY KEY AUTOINCREMENT,
    Loc_Name TEXT UNIQUE NOT NULL
);
```

Course:

```
CREATE TABLE Course (
    Crs_ID INTEGER NOT NULL
        PRIMARY KEY AUTOINCREMENT,
    Crs_Subject TEXT NOT NULL CHECK (length(Crs_Subject) <= 4),
    Crs_CatalogNumber TEXT NOT NULL
        CHECK (Crs_CatalogNumber >= 0 and Crs_CatalogNumber <= 9999),
    UNIQUE(Crs_Subject, Crs_CatalogNumber) ON CONFLICT ABORT
);
```

CourseTopic:

```
CREATE TABLE CourseTopic (
    CrsTopic_ID INTEGER NOT NULL
        PRIMARY KEY AUTOINCREMENT,
    Crs_ID INTEGER NOT NULL
        REFERENCES Course(Crs_ID) ON DELETE CASCADE,
    CrsTopic_Name TEXT NOT NULL,
    UNIQUE(Crs_ID, CrsTopic_Name) ON CONFLICT ABORT
);
```

Class:

```
CREATE TABLE Class (
    Class_ID INTEGER NOT NULL
        PRIMARY KEY AUTOINCREMENT,
    Crs_ID INTEGER NOT NULL
        REFERENCES Course(Crs_ID) ON DELETE CASCADE,
    Sem_ID INTEGER NOT NULL
        REFERENCES Semester(Sem_ID) ON DELETE CASCADE,
        UNIQUE (Crs_ID, Sem_ID) ON CONFLICT ABORT
);
```

Registration:

```
CREATE TABLE Registration (
    Reg_ID INTEGER NOT NULL
        PRIMARY KEY AUTOINCREMENT,
    Class_ID INTEGER NOT NULL
        REFERENCES Class(Class_ID) ON DELETE CASCADE,
    UnivMem_ID INTEGER NOT NULL
        REFERENCES UniversityMember(UnivMem_ID) ON DELETE CASCADE,
    Reg_Type TEXT NOT NULL
        CHECK (Reg_Type in ('TA', 'STUD', 'PROF')),
        UNIQUE (Class_ID, UnivMem_ID) ON CONFLICT ABORT
);
```

Assignment:

```
CREATE TABLE Assignment (
    Assign_ID INTEGER NOT NULL
        PRIMARY KEY AUTOINCREMENT,
    Class_ID INTEGER NOT NULL
        REFERENCES Class(Class_ID) ON DELETE CASCADE,
    Assign_Name TEXT NOT NULL,
        UNIQUE (Class_ID, Assign_Name) ON CONFLICT ABORT
);
```

AssignmentTopic:

```
CREATE TABLE AssignmentTopic (
    AssignTopic_ID INTEGER NOT NULL
        PRIMARY KEY AUTOINCREMENT,
    Assign_ID INTEGER NOT NULL
        REFERENCES Assignment(Assign_ID) ON DELETE CASCADE,
    CrsTopic_ID INTEGER NOT NULL
        REFERENCES CourseTopic(CrsTopic_ID) ON DELETE CASCADE,
    UNIQUE (Assign_ID, CrsTopic_ID) ON CONFLICT ABORT
);
```

OfficeHourSession:

```
CREATE TABLE OfficeHourSession (
   OHS_ID INTEGER NOT NULL
      PRIMARY KEY AUTOINCREMENT,
   Class_ID INTEGER NOT NULL
      REFERENCES Class(Class_ID) ON DELETE CASCADE,
   UnivMem_ID INTEGER NOT NULL
      REFERENCES Instructor(UnivMem_ID) ON DELETE CASCADE,
   Loc_ID INTEGER NOT NULL
      REFERENCES Location(Loc_ID) ON DELETE CASCADE,
   OHS_StartDateTime DATE NOT NULL,
   OHS_EndDateTime DATE NOT NULL,
   OHS_Status TEXT NOT NULL,
   CHECK (OHS_EndDateTime > OHS_StartDateTime)
);
```

OfficeHourQuestion:

```
CREATE TABLE OfficeHourQuestion (
   OHQ_ID INTEGER NOT NULL
      PRIMARY KEY AUTOINCREMENT,
   OHS_ID INTEGER NOT NULL
      REFERENCES OfficeHourSession(OHS_ID) ON DELETE CASCADE,
   Assign_ID INTEGER
      REFERENCES Assignment(Assign_ID) ON DELETE CASCADE,
   UnivMem_ID INTEGER NOT NULL
      REFERENCES Student(UnivMem_ID) ON DELETE CASCADE,
   OHQ_Status TEXT NOT NULL,
   OHQ_StudentComment TEXT NOT NULL,
   OHQ_InstructorComment TEXT NOT NULL,
   OHQ_OpenedAt DATE NOT NULL,
   OHQ_ClosedAt DATE NOT,
);
```

OfficeHourQuestionTopic:

```
CREATE TABLE OfficeHourQuestionTopic (
   OHQT_ID INTEGER NOT NULL
      PRIMARY KEY AUTOINCREMENT,
   OHQ_ID INTEGER NOT NULL
      REFERENCES OfficeHourQuestion(OHQ_ID),
   CrsTopic_ID INTEGER NOT NULL
      REFERENCES CourseTopic(CrsTopic_ID),
      UNIQUE (OHQ_ID, CrsTopic_ID) ON CONFLICT ABORT
);
```

CREATE TRIGGER QUERIES:

```
CREATE TRIGGER IF NOT EXISTS VerifyInstructorOnOHSInsert
   BEFORE INSERT ON OfficeHourSession
   WHEN NOT EXISTS(
      SELECT 1 FROM Registration R
         WHERE R.UnivMem_ID=NEW.UnivMem_ID
            AND R.Reg_Type in ('PROF', 'TA')
            AND R.Class_ID=NEW.Class_ID
   )
   BEGIN
      SELECT RAISE(ABORT, 'Error the UniversityMember must be registered as either a
professor or a TA in that class to hold office hours.');
   END;

CREATE TRIGGER IF NOT EXISTS VerifyInstructorOnOHSUpdate
   BEFORE UPDATE ON OfficeHourSession
   WHEN NOT EXISTS(
      SELECT 1 FROM Registration R
         WHERE R.UnivMem_ID=NEW.UnivMem_ID
            AND R.Reg_Type in ('PROF', 'TA')
            AND R.Class_ID=NEW.Class_ID
   )
   BEGIN
      SELECT RAISE(ABORT, 'Error the UniversityMember must be registered as either a
professor or a TA in that class to hold office hours.');
   END;


CREATE TRIGGER IF NOT EXISTS VerifyStudentOnOHQUpdate
   BEFORE UPDATE ON OfficeHourQuestion
   WHEN NOT EXISTS(
      SELECT 1 FROM Registration R
         JOIN OfficeHourSession OHS ON OHS.OHS_ID = NEW.OHS_ID
         WHERE R.UnivMem_ID=NEW.UnivMem_ID
            AND R.Reg_Type == 'STUD'
            AND R.Class_ID=OHS.Class_ID
   )
   BEGIN
      SELECT RAISE(ABORT, 'Error the UniversityMember must be registered as a student in
that class to ask a question.');
   END;

CREATE TRIGGER IF NOT EXISTS VerifyStudentOnOHQInsert
   BEFORE INSERT ON OfficeHourQuestion
```

```
    WHEN NOT EXISTS(
        SELECT 1 FROM Registration R
            JOIN OfficeHourSession OHS ON OHS.OHS_ID = NEW.OHS_ID
            WHERE R.UnivMem_ID=NEW.UnivMem_ID
                AND R.Reg_Type == 'STUD'
                AND R.Class_ID=OHS.Class_ID
    )
    BEGIN
        SELECT RAISE(ABORT, 'Error the UniversityMember must be registered as a student in
that class to ask a question.');
    END;
```

INSERTION QUERIES:

University Member:

```
INSERT INTO UniversityMember (
    UnivMem_FirstName,
    UnivMem_LastName,
    UnivMem_CompID,
    UnivMem_IsStudent,
    UnivMem_IsInstructor
) VALUES
    ('Jonah', 'Kim', 'crd3vm', TRUE, TRUE),
    ('Yuxi', 'Yang', 'yy6znb', TRUE, FALSE),
    ('Tammy', 'Ngo', 'bsy6pq', TRUE, FALSE),
    ('Mary', 'Smith', 'mls4aa', FALSE, TRUE),
    ('Ishan', 'Mathur', 'anm3hh', FALSE, TRUE),
    ('John', 'Doe', 'aaa1aa', TRUE, TRUE),
    ('Jane', 'Doe', 'jnd6a', TRUE, FALSE),
    ('Richard', 'Ngyuen', 'rcng1', FALSE, TRUE),
    ('William', 'McBurney', 'wb288', FALSE, TRUE);
```

Student:

```
INSERT INTO Student
    (UnivMem_ID)
    VALUES (1), (2), (3), (6), (7);
```

Instructor:

```
INSERT INTO Instructor
    (UnivMem_ID, Instructor_Type)
    VALUES
        (1, 'TA'),
        (4, 'PROF'),
        (5, 'TA'),
        (6, 'PROF'),
        (8, 'PROF'),
        (9, 'PROF');
```

Professor:

```
INSERT INTO Professor
    (UnivMem_ID)
    VALUES
        (4),
        (6),
        (8),
        (9);
```

TeachingAssistant:

```
INSERT INTO TeachingAssistant
    (UnivMem_ID)
    VALUES
        (1),
        (5);
```

Semester:

```
INSERT INTO Semester
    (Sem_Name, Sem_Year)
    VALUES
        ('SPRING', 2023),
        ('SUMMER', 2023),
        ('FALL', 2023);
```

Location:

```
INSERT INTO Location
    (Loc_Name)
    VALUES
        ('Online'),
        ('Rice 130'),
        ('Thornton Stacks');
```

Course:

```
INSERT INTO Course
    (Crs_Subject, Crs_CatalogNumber)
    VALUES
        ('CS', '3140'),
        ('APMA', '3100'),
        ('CS', '4750');
```

CourseTopic:

```
INSERT INTO CourseTopic
    (Crs_ID, CrsTopic_Name)
    VALUES
        (1, 'Design Patterns'),
        (1, 'Architecture'),
        (1, 'JDBC'),
        (1, 'General'),
        (1, 'Logistics'),
        (2, 'Hypothesis Testing'),
        (2, 'Combinatorics'),
        (2, 'Probability Distributions'),
        (3, 'SQL'),
        (3, 'Normalization'),
        (3, 'General'),
        (3, 'Logistics'),
        (3, 'Project'),
        (3, 'Data Redundancy'),
        (3, 'Specialization Hierarchies'),
        (1, 'GitHub'),
        (1, 'JavaFX');
```

Class:

```
INSERT INTO Class
    (Crs_ID, Sem_ID)
    VALUES
        (1, 1),
        (1, 3),
        (2, 2),
        (3, 2);
```

Registration:

```
INSERT INTO Registration
     (Class_ID, UnivMem_ID, Reg_Type)
     VALUES
         (1, 1, 'TA'), -- Jonah
         (2, 1, 'TA'), -- Jonah
         (1, 8, 'PROF'), -- Richard Nguyen
         (2, 9, 'PROF'), -- William McBurney
         (2, 6, 'STUD'), -- John Doe
         (3, 1, 'STUD'),
         (3, 2, 'STUD'),
         (3, 3, 'STUD'),
         (4, 4, 'PROF'), -- Mary Smith
         (4, 5, 'TA'), -- Ishan
         (3, 6, 'STUD'), -- John Doe
         (3, 7, 'STUD'), -- Jane Doe
         (1, 6, 'STUD'), -- John Doe
         (4, 1, 'STUD'),-- Jonah
         (1, 2, 'STUD'),--Yuxi
         (1, 7, 'STUD'); --Jane Doe
```

Assignment:

```
INSERT INTO Assignment
   (Class_ID, Assign_Name)
   VALUES
      (1, 'HW1 A/B Apportionment'),
      (1, 'HW1 C Apportionment'),
      (1, 'HW2 Wordle Debugging'),
      (1, 'HW3 Apportionment Refactoring'),
      (2, 'HW1 A/B Apportionment'),
      (2, 'HW1 C Apportionment'),
      (2, 'HW2 Wordle Debugging'),
      (2, 'HW3 Apportionment Refactoring'),
      (4, 'Practice Chapter 6'),
      (4, 'Practice Chapter 7'),
      (4, 'Project Report 1'),
      (4, 'Project Report 2'),
      (4, 'Project Final Report'),
      (4, 'Quiz 1'),
      (4, 'Quiz 2');
```

AssignmentTopic:

```
INSERT INTO AssignmentTopic
   (Assign_ID, CrsTopic_ID)
   VALUES
      (1,4),
      (2, 1),
      (2, 4),
      (3, 1),
      (4, 1),
      (4, 2),
      (5, 4),
      (6, 1),
      (6, 4),
      (7, 1),
      (8, 1),
      (8, 2),
      (9, 10),
      (10, 9),
      (11, 3),
      (12, 13),
      (13, 13),
      (13, 14),
      (13, 15),
      (14, 11),
      (14, 12),
      (15, 11),
      (15, 12);
```

OfficeHourSession:

```
INSERT INTO OfficeHourSession
  (
    Class_ID,
    UnivMem_ID,
    Loc_ID,
    OHS_StartDateTime,
    OHS_EndDateTime,
    OHS_Status
  )
  VALUES
    (4, 5, 1, '2023-06-29 13:00:00', '2023-06-29 15:00:00', 'OPEN'),
    (1, 1, 1, '2023-02-03 10:00:00', '2023-02-03 12:30:00', 'OPEN'), --Jonah CS 3140
    (1, 1, 1, '2023-03-03 10:00:00', '2023-03-03 14:30:00', 'OPEN'); --Jonah CS 3140
```

OfficeHourQuestion:

```
INSERT INTO OfficeHourQuestion
  (
    OHS_ID,
    Assign_ID,
    UnivMem_ID,
    OHQ_Status,
    OHQ_StudentComment,
    OHQ_InstructorComment,
    OHQ_OpenedAt,
    OHQ_ClosedAt
  )
  VALUES
```

```
    (1, NULL, 1, 'PENDING', 'Wanted clarification on a topic', 'The student wanted help since
the textbook did not explain it very well', '2023-06-29 13:00:00', NULL),
    (2, 1, 2, 'ANSWERED', 'How to resolve conflict on Github', 'The issue was experiencing an
issue with their repository where the name was already taken.', '2023-02-03 10:03:00',
'2023-02-03 10:14:00'),
    (2, 1, 2, 'UNANSWERED', 'There are issues with my repository on Github', 'They did not
name their repository correctly.', '2023-02-03 10:10:00', '2023-02-03 10:18:00'),
    (2, 8, 6, 'ANSWERED', 'I updated my mac to the latest java version but when running the
initial HW3.java file I am getting the error. How to resolve it?', 'They needed to install an updated
jdk', '2023-02-03 10:12:00', '2023-02-03 10:20:00'),
    (3, 7, 7, 'ANSWERED', 'JavaFX Runtime library issues', 'They needed the M1 JavaFX
library for Mac.', '2023-03-03 10:03:00', '2023-03-03 10:10:00'),
    (3, 8, 7, 'UNANSWERED', 'How to create build dependencies?', 'They wanted help with
gradle.build', '2023-03-03 10:06:00', '2023-03-03 10:10:00');
```

OfficeHourQuestionTopic:

```
INSERT INTO OfficeHourQuestionTopic
  (OHQ_ID, CrsTopic_ID)
  VALUES
    (1, 11),
    (2, 16),
    (3, 16),
    (4, 4),
    (5, 4),
    (6,4);
```

SELECT ALL QUERIES:

```
SELECT * FROM UniversityMember;
SELECT * FROM Student;
SELECT * FROM Instructor;
SELECT * FROM Professor;
SELECT * FROM TeachingAssistant;
SELECT * FROM Semester;
SELECT * FROM Location;
SELECT * FROM Course;
SELECT * FROM CourseTopic;
SELECT * FROM Class;
SELECT * FROM Registration;
SELECT * FROM Assignment;
SELECT * FROM AssignmentTopic;
SELECT * FROM OfficeHourSession;
SELECT * FROM OfficeHourQuestion;
SELECT * FROM OfficeHourQuestionTopic;
```

DROP ALL QUERIES:

These queries are ordered so that you can drop all tables:

```
DROP TABLE if exists OfficeHourQuestionTopic;
DROP TABLE if exists OfficeHourQuestion;
DROP TABLE if exists OfficeHourSession;
DROP TABLE if exists AssignmentTopic;
DROP TABLE if exists Assignment;
DROP TABLE if exists CourseTopic;
DROP TABLE if exists Registration;
DROP TABLE if exists Class;

DROP TABLE if exists Course;
DROP TABLE if exists Location;
DROP TABLE if exists Semester;
DROP TABLE if exists Professor;
DROP TABLE if exists TeachingAssistant;
DROP TABLE if exists Instructor;
DROP TABLE if exists Student;
DROP TABLE if exists UniversityMember;
```

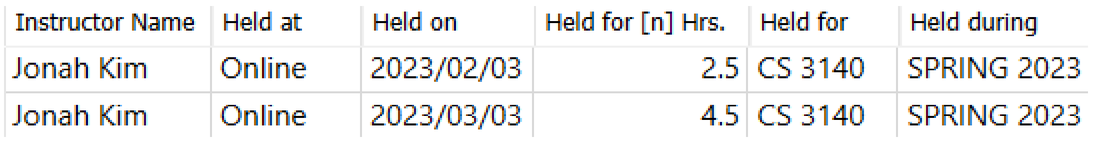
CRITERIA BASED SELECTS:

OfficeHourSessions by Duration:

```
SELECT
    UM.UnivMem_FirstName || ' ' || UM.UnivMem_LastName as 'Instructor Name',
    L.Loc_Name as 'Held at',
    strftime('%Y/%m/%d', OHS.OHS_StartDateTime) as 'Held on',
    (strftime('%s', OHS.OHS_EndDateTime) - strftime('%s', OHS.OHS_StartDateTime)) /
3600.0 AS 'Held for [n] Hrs.',
    CRS.Crs_Subject || ' ' || CRS.Crs_CatalogNumber as 'Held for',
    S.Sem_Name || ' ' || S.Sem_Year as 'Held during'
  FROM OfficeHourSession OHS
  JOIN Location L USING (Loc_ID)
  JOIN UniversityMember UM USING (UnivMem_ID)
  JOIN Class CLS USING (Class_ID)
  JOIN Course CRS USING (Crs_ID)
  JOIN Semester S USING (Sem_ID)
  WHERE
    Loc_Name LIKE '%online%' AND
    "Held for [n] Hrs." > 2;
```

Table 1

Lists all office hours that were held online for more than 2 hours

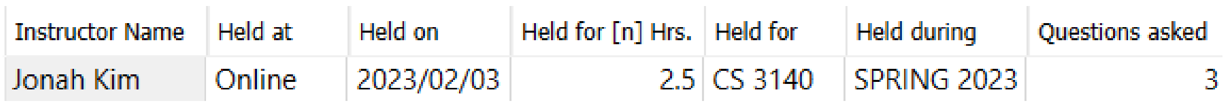


Instructor Name	Held at	Held on	Held for [n] Hrs.	Held for	Held during
Jonah Kim	Online	2023/02/03	2.5	CS 3140	SPRING 2023
Jonah Kim	Online	2023/03/03	4.5	CS 3140	SPRING 2023

OfficeHourSessions by Questions Asked:

```
SELECT
    UM.UnivMem_FirstName || ' ' || UM.UnivMem_LastName as 'Instructor Name',
    L.Loc_Name as 'Held at',
    strftime('%Y/%m/%d', OHS.OHS_StartDateTime) as 'Held on',
    (strftime('%s', OHS.OHS_EndDateTime) - strftime('%s', OHS.OHS_StartDateTime)) /
3600.0 AS 'Held for [n] Hrs.',
    CRS.Crs_Subject || ' ' || CRS.Crs_CatalogNumber as 'Held for',
    S.Sem_Name || ' ' || S.Sem_Year as 'Held during',
    "Questions asked"
  FROM OfficeHourSession OHS
  JOIN (
    SELECT OHS_ID, COUNT(*) as 'Questions asked' FROM OfficeHourQuestion GROUP BY
OHS_ID
  ) USING (OHS_ID)
  JOIN UniversityMember UM USING (UnivMem_ID)
  JOIN Class CLS USING (Class_ID)
  JOIN Location L USING (Loc_ID)
  JOIN Course CRS USING (Crs_ID)
  JOIN Semester S USING (Sem_ID)
  WHERE "Questions asked" > 2;
```

Table 2
Lists all office hours where there were more than 2 (exclusive) questions asked



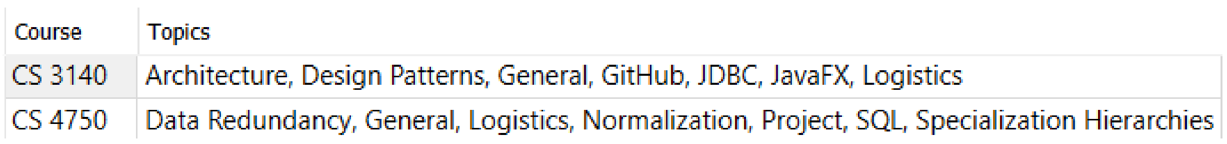
Instructor Name	Held at	Held on	Held for [n] Hrs.	Held for	Held during	Questions asked
Jonah Kim	Online	2023/02/03	2.5	CS 3140	SPRING 2023	3

CourseTopics for CS

```
SELECT
    CRS.Crs_Subject || ' ' || CRS.Crs_CatalogNumber as 'Course',
    group_concat(CT.CrsTopic_Name, ', ') as 'Topics'
  FROM CourseTopic CT
  JOIN Course CRS USING (Crs_ID)
  WHERE Crs_Subject LIKE 'CS'
```

```
   GROUP BY Crs_ID;
```

Table 3
Lists all topics for CS courses

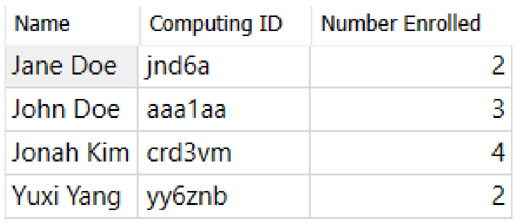


Course	Topics
CS 3140	Architecture, Design Patterns, General, GitHub, JDBC, JavaFX, Logistics
CS 4750	Data Redundancy, General, Logistics, Normalization, Project, SQL, Specialization Hierarchies

UniversityMembers by Enrollment Count:

```
SELECT
      UM.UnivMem_FirstName || ' ' || UM.UnivMem_LastName as 'Name',
      UM.UnivMem_CompID as 'Computing ID',
      count(UnivMem_ID) as 'Number Enrolled'
   FROM Registration R
   JOIN UniversityMember UM USING (UnivMem_ID)
   GROUP BY UnivMem_ID
   HAVING "Number Enrolled" > 1
   ORDER BY UnivMem_FirstName, UnivMem_LastName;
```

Table 4
Lists all university members who are enrolled in more than one course



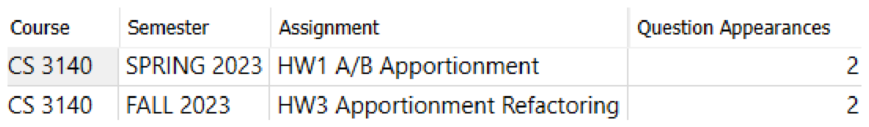
Name	Computing ID	Number Enrolled
Jane Doe	jnd6a	2
John Doe	aaa1aa	3
Jonah Kim	crd3vm	4
Yuxi Yang	yy6znb	2

Assignments by Question Appearances:

```
SELECT
      CRS.Crs_Subject || ' ' || CRS.Crs_CatalogNumber as 'Course',
      S.Sem_Name || ' ' || S.Sem_Year as 'Semester',
      Assign_Name as 'Assignment',
      COUNT(OHQ_ID) as 'Question Appearances'
   FROM Assignment
   JOIN OfficeHourQuestion USING (Assign_ID)
```

```
  JOIN Class CLS USING (Class_ID)
  JOIN Course CRS USING (Crs_ID)
  JOIN Semester S USING (Sem_ID)
  GROUP BY Assign_ID
  HAVING "Question Appearances" > 1;
```

Table 5
Lists all assignments which were linked more than once in office hour questions



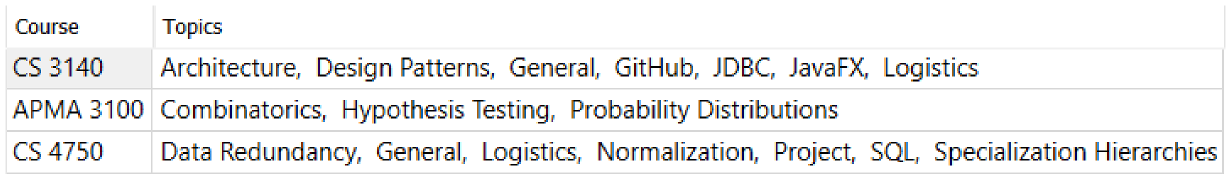
Course	Semester	Assignment	Question Appearances
CS 3140	SPRING 2023	HW1 A/B Apportionment	2
CS 3140	FALL 2023	HW3 Apportionment Refactoring	2

SUMMARIZATION SELECTS:

All CourseTopics:

```
SELECT
    CRS.Crs_Subject || ' ' || CRS.Crs_CatalogNumber as 'Course',
    group_concat(CT.CrsTopic_Name, ',  ') as 'Topics'
  FROM CourseTopic CT
  JOIN Course CRS Using(Crs_ID)
  GROUP BY CT.Crs_ID;
```

Table 6
Shows all the CourseTopics for a course

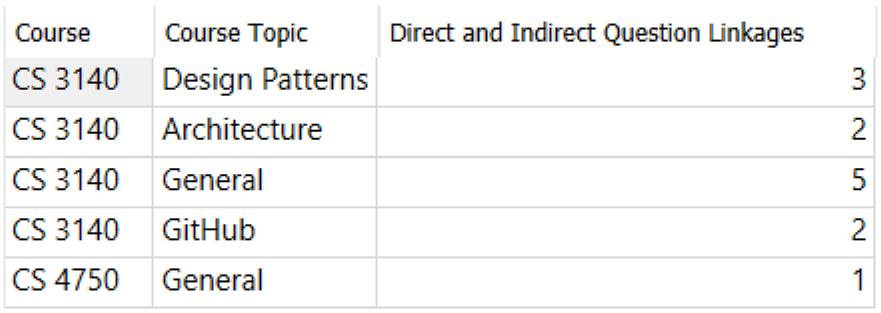


Course	Topics
CS 3140	Architecture, Design Patterns, General, GitHub, JDBC, JavaFX, Logistics
APMA 3100	Combinatorics, Hypothesis Testing, Probability Distributions
CS 4750	Data Redundancy, General, Logistics, Normalization, Project, SQL, Specialization Hierarchies

CourseTopic Appearances:

```
SELECT
    CRS.Crs_Subject || ' ' || CRS.Crs_CatalogNumber as 'Course',
    CT.CrsTopic_Name as "Course Topic",
    SUM(OCCURRENCES) as 'Direct and Indirect Question Linkages'
  FROM (
    SELECT
        StudentLinkedTopics.CrsTopic_ID,
        Count(DISTINCT OHQ.OHQ_ID) as OCCURRENCES
      FROM OfficeHourQuestion OHQ
      JOIN OfficeHourQuestionTopic QT USING (OHQ_ID)
      JOIN CourseTopic StudentLinkedTopics ON
StudentLinkedTopics.CrsTopic_ID=QT.CrsTopic_ID
      GROUP BY StudentLinkedTopics.CrsTopic_ID
    UNION
    SELECT
        AssignmentLinkedTopics.CrsTopic_ID,
        Count(DISTINCT OHQ.OHQ_ID) as OCCURRENCES
      FROM OfficeHourQuestion OHQ
      JOIN Assignment A USING (Assign_ID)
      JOIN AssignmentTopic AT USING (Assign_ID)
      JOIN CourseTopic AssignmentLinkedTopics ON
AssignmentLinkedTopics.CrsTopic_ID=AT.CrsTopic_ID
      GROUP BY AssignmentLinkedTopics.CrsTopic_ID
  )
  JOIN CourseTopic CT USING (CrsTopic_ID)
  JOIN Course CRS USING (Crs_ID)
  GROUP BY CrsTopic_ID
  ORDER BY "Course";
```

Table 7
Shows the cumulative number of times a course topic has appeared in office hour questions (removing those which do not appear).



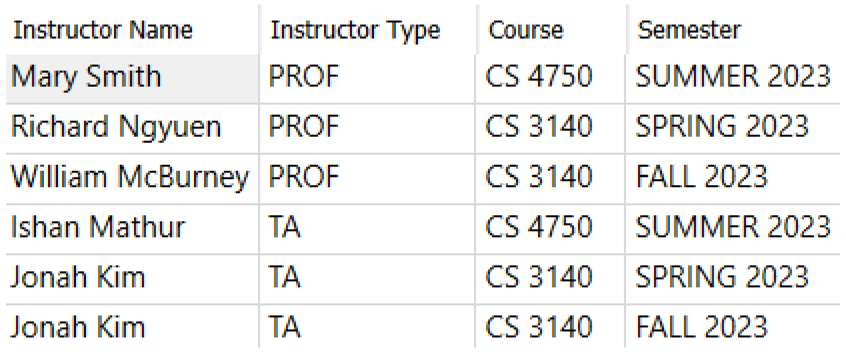
Course	Course Topic	Direct and Indirect Question Linkages
CS 3140	Design Patterns	3
CS 3140	Architecture	2
CS 3140	General	5
CS 3140	GitHub	2
CS 4750	General	1

All Instructors:

```
SELECT
    U.UnivMem_FirstName || ' ' || U.UnivMem_LastName as 'Instructor Name',
    R.Reg_Type as 'Instructor Type',
    Crs.Crs_Subject || ' ' || Crs.Crs_CatalogNumber as 'Course',
    S.Sem_Name || ' ' || S.Sem_Year as 'Semester'
  FROM Registration R
  JOIN UniversityMember U USING (UnivMem_ID)
  JOIN Class Cls USING (Class_ID)
```

```
   JOIN Course Crs USING (Crs_ID)
   JOIN Semester S USING (Sem_ID)
   WHERE R.Reg_Type in ('PROF', 'TA')
   ORDER BY R.Reg_Type, U.UnivMem_FirstName, U.UnivMem_LastName;
```

Table 8
List All Instructors

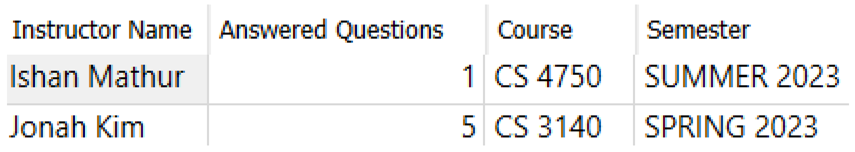


Instructor Name	Instructor Type	Course	Semester
Mary Smith	PROF	CS 4750	SUMMER 2023
Richard Ngyuen	PROF	CS 3140	SPRING 2023
William McBurney	PROF	CS 3140	FALL 2023
Ishan Mathur	TA	CS 4750	SUMMER 2023
Jonah Kim	TA	CS 3140	SPRING 2023
Jonah Kim	TA	CS 3140	FALL 2023

Questions Answered by Instructor:

```
SELECT
      UM.UnivMem_FirstName || ' ' || UM.UnivMem_LastName as 'Instructor Name',
      COUNT(OHQ_ID) as 'Answered Questions',
      Crs.Crs_Subject || ' ' || Crs.Crs_CatalogNumber as 'Course',
      S.Sem_Name || ' ' || S.Sem_Year as 'Semester'
   FROM OfficeHourSession OHS
   JOIN OfficeHourQuestion OHQ USING (OHS_ID)
   JOIN UniversityMember UM Using (UnivMem_ID)
   JOIN Class CLS USING (Class_ID)
   JOIN Course CRS USING (Crs_ID)
   JOIN Semester S USING (Sem_ID)
   GROUP BY CLS.Class_ID
   ORDER BY UM.UnivMem_FirstName, UM.UnivMem_LastName;
```

Table 9
Shows the number of questions answered per instructor for a particular class

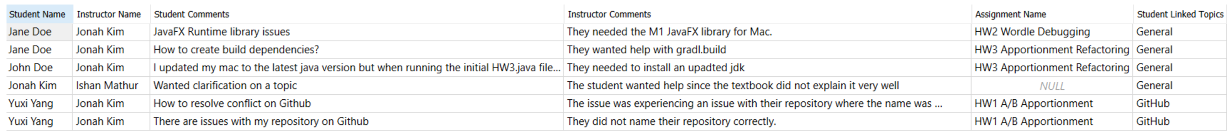


Instructor Name	Answered Questions	Course	Semester
Ishan Mathur	1	CS 4750	SUMMER 2023
Jonah Kim	5	CS 3140	SPRING 2023

All OfficeHourQuestion

```
SELECT
      Student.UnivMem_FirstName || ' ' || Student.UnivMem_LastName as 'Student Name',
      Instructor.UnivMem_FirstName || ' ' || Instructor.UnivMem_LastName as 'Instructor Name',
      OHQ.OHQ_StudentComment as 'Student Comments',
      OHQ.OHQ_InstructorComment as 'Instructor Comments',
      A.Assign_Name as 'Assignment Name',
      group_concat(CrsTopic_Name, ', ') as 'Student Linked Topics'
   FROM OfficeHourQuestion OHQ
   JOIN OfficeHourSession OHS USING (OHS_ID)
   JOIN UniversityMember Student ON Student.UnivMem_ID=OHQ.UnivMem_ID
   JOIN UniversityMember Instructor ON Instructor.UnivMem_ID=OHS.UnivMem_ID
   LEFT JOIN Assignment A USING (Assign_ID)
   LEFT JOIN OfficeHourQuestionTopic OHQT USING (OHQ_ID)
   LEFT JOIN CourseTopic CT USING (CrsTopic_ID)
   GROUP BY OHQ_ID
   ORDER BY "Student Name", "Instructor Name";
```

Table 10
Shows information about OfficeHourQuestions: Student, Instructor, question comments, Assignment, and student linked topics



Student Name	Instructor Name	Student Comments	Instructor Comments	Assignment Name	Student Linked Topics
Jane Doe	Jonah Kim	JavaFX Runtime library issues	They needed the M1 JavaFX library for Mac.	HW2 Wordle Debugging	General
Jane Doe	Jonah Kim	How to create build dependencies?	They wanted help with gradl.build	HW3 Apportionment Refactoring	General
John Doe	Jonah Kim	I updated my mac to the latest java version but when running the initial HW3.java file...	They needed to install an upadted jdk	HW3 Apportionment Refactoring	General
Jonah Kim	Ishan Mathur	Wanted clarification on a topic	The student wanted help since the textbook did not explain it very well	NULL	General
Yuxi Yang	Jonah Kim	How to resolve conflict on Github	The issue was experiencing an issue with their repository where the name was ...	HW1 A/B Apportionment	GitHub
Yuxi Yang	Jonah Kim	There are issues with my repository on Github	They did not name their repository correctly.	HW1 A/B Apportionment	GitHub

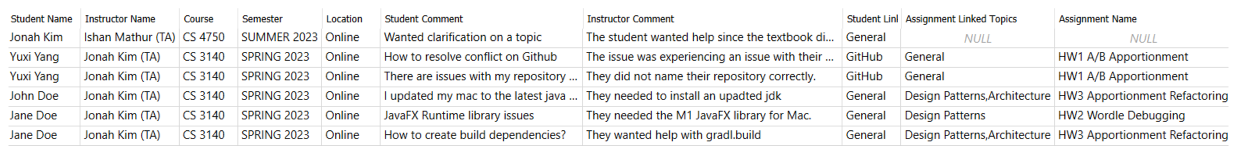
MASTER SELECT:

Question Summarization:

```
SELECT
      Student.UnivMem_FirstName || ' ' || Student.UnivMem_LastName as 'Student Name',
      Instructor.UnivMem_FirstName || ' ' || Instructor.UnivMem_LastName || ' (' || R.Reg_Type ||
')' as 'Instructor Name',
      CRS.Crs_Subject || ' ' || CRS.Crs_CatalogNumber as 'Course',
      S.Sem_Name || ' ' || S.Sem_Year as 'Semester',
      L.Loc_Name as 'Location',
      OHQ.OHQ_StudentComment as 'Student Comment',
      OHQ.OHQ_InstructorComment as 'Instructor Comment',
      SUBQUERY1.STUD_LINKED_TOPICS as 'Student Linked Topics',
      SUBQUERY2.ASSIGN_LINKED_TOPICS as 'Assignment Linked Topics',
      SUBQUERY2.Assign_Name as 'Assignment Name'
```

```
    FROM OfficeHourQuestion OHQ
    JOIN OfficeHourSession OHS USING (OHS_ID)
    JOIN Location L USING (Loc_ID)
    JOIN UniversityMember Instructor ON Instructor.UnivMem_ID=OHS.UnivMem_ID
    JOIN UniversityMember Student ON Student.UnivMem_ID=OHQ.UnivMem_ID
    JOIN Class CLS USING (Class_ID)
    JOIN Course CRS USING (Crs_ID)
    JOIN Semester S USING (Sem_ID)
    LEFT JOIN (
        SELECT
                OHQ_SUBQUERY.OHQ_ID,
                group_concat(DISTINCT CT.CrsTopic_Name) as STUD_LINKED_TOPICS
            FROM OfficeHourQuestion OHQ_SUBQUERY
            JOIN OfficeHourQuestionTopic QT USING(OHQ_ID)
            JOIN CourseTopic CT USING(CrsTopic_ID)
            GROUP BY OHQ_SUBQUERY.OHQ_ID
        ) SUBQUERY1 ON SUBQUERY1.OHQ_ID=OHQ.OHQ_ID
     LEFT JOIN (
        SELECT
                OHQ_SUBQUERY.OHQ_ID,
                QA.Assign_Name,
                group_concat(DISTINCT CT.CrsTopic_Name) as ASSIGN_LINKED_TOPICS
            FROM OfficeHourQuestion OHQ_SUBQUERY
            JOIN Assignment QA USING(Assign_ID)
            JOIN AssignmentTopic QAT USING (Assign_ID)
            JOIN CourseTopic CT USING(CrsTopic_ID)
            GROUP BY OHQ_SUBQUERY.OHQ_ID
        ) SUBQUERY2 ON SUBQUERY2.OHQ_ID=OHQ.OHQ_ID
    JOIN Registration R ON R.UnivMem_ID=Instructor.UnivMem_ID AND
R.Class_ID=OHS.Class_ID;
```

Table 11
Lists all Questions: their student, their instructor, their course, their semester, their location, their comments, and their topics (12 tables joined out of 16)



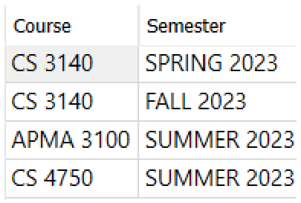
Student Name	Instructor Name	Course	Semester	Location	Student Comment	Instructor Comment	Student Linl	Assignment Linked Topics	Assignment Name
Jonah Kim	Ishan Mathur (TA)	CS 4750	SUMMER 2023	Online	Wanted clarification on a topic	The student wanted help since the textbook di...	General	NULL	NULL
Yuxi Yang	Jonah Kim (TA)	CS 3140	SPRING 2023	Online	How to resolve conflict on Github	The issue was experiencing an issue with their ...	GitHub	General	HW1 A/B Apportionment
Yuxi Yang	Jonah Kim (TA)	CS 3140	SPRING 2023	Online	There are issues with my repository ...	They did not name their repository correctly.	GitHub	General	HW1 A/B Apportionment
John Doe	Jonah Kim (TA)	CS 3140	SPRING 2023	Online	I updated my mac to the latest java ...	They needed to install an upadted jdk	General	Design Patterns,Architecture	HW3 Apportionment Refactoring
Jane Doe	Jonah Kim (TA)	CS 3140	SPRING 2023	Online	JavaFX Runtime library issues	They needed the M1 JavaFX library for Mac.	General	Design Patterns	HW2 Wordle Debugging
Jane Doe	Jonah Kim (TA)	CS 3140	SPRING 2023	Online	How to create build dependencies?	They wanted help with gradl.build	General	Design Patterns,Architecture	HW3 Apportionment Refactoring

ASSOCIATIVE ENTITY SELECTS:

Class:

```
SELECT
    CRS.Crs_Subject || ' ' || CRS.Crs_CatalogNumber as 'Course',
    S.Sem_Name || ' ' || S.Sem_Year as 'Semester'
  FROM Class CLS
  JOIN Course CRS USING (Crs_ID)
  JOIN Semester S USING (Sem_ID);
```

Table 12
Shows course and semester

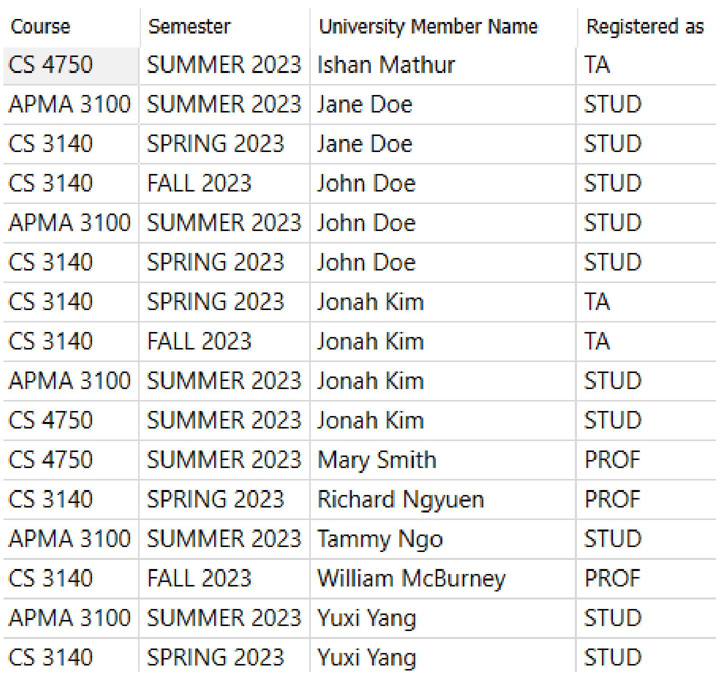


Course	Semester
CS 3140	SPRING 2023
CS 3140	FALL 2023
APMA 3100	SUMMER 2023
CS 4750	SUMMER 2023

Registration:

```
SELECT
    UM.UnivMem_FirstName || ' ' || UM.UnivMem_LastName as 'University Member Name',
    R.Reg_Type as 'Registered as',
    CRS.Crs_Subject || ' ' || CRS.Crs_CatalogNumber || ' ' || S.Sem_Name || ' ' || S.Sem_Year
as 'Registered in'
  FROM Registration R
  JOIN UniversityMember UM USING (UnivMem_ID)
  JOIN Class CLS USING (Class_ID)
  JOIN Course CRS USING (Crs_ID)
  JOIN Semester S USING (Sem_ID)
  ORDER BY "University Member Name";
```

Table 13
Shows UniversityMember, registration type, and class

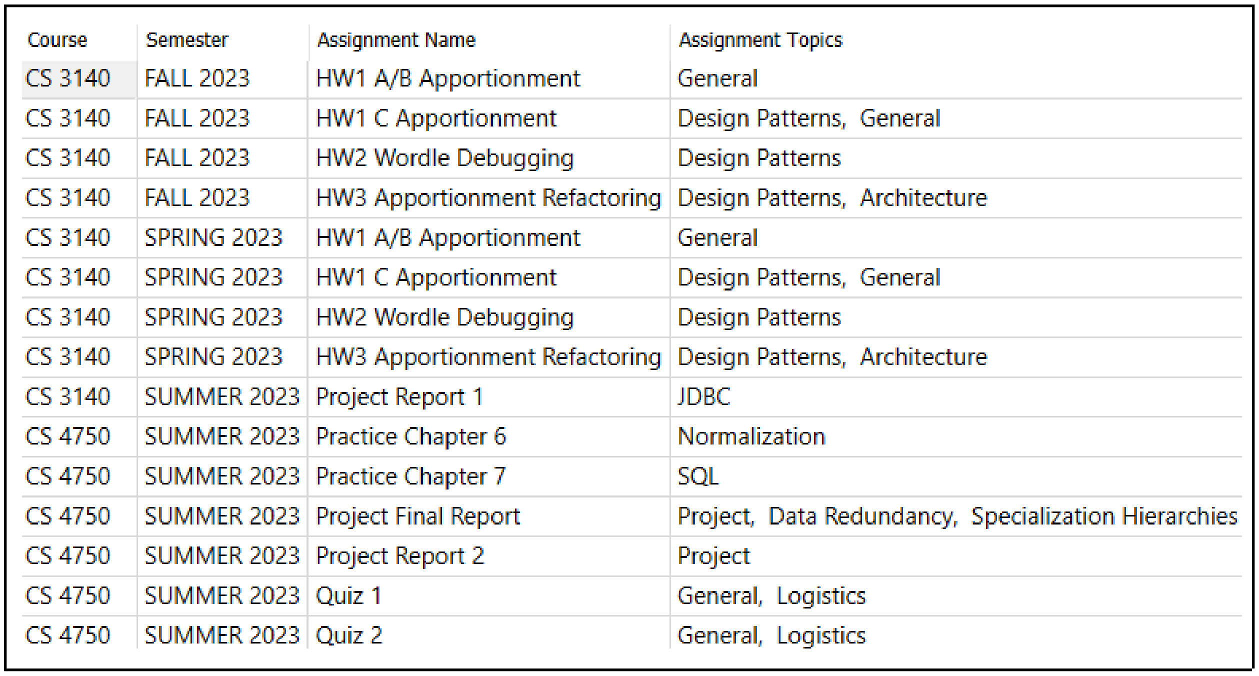


Course	Semester	University Member Name	Registered as
CS 4750	SUMMER 2023	Ishan Mathur	TA
APMA 3100	SUMMER 2023	Jane Doe	STUD
CS 3140	SPRING 2023	Jane Doe	STUD
CS 3140	FALL 2023	John Doe	STUD
APMA 3100	SUMMER 2023	John Doe	STUD
CS 3140	SPRING 2023	John Doe	STUD
CS 3140	SPRING 2023	Jonah Kim	TA
CS 3140	FALL 2023	Jonah Kim	TA
APMA 3100	SUMMER 2023	Jonah Kim	STUD
CS 4750	SUMMER 2023	Jonah Kim	STUD
CS 4750	SUMMER 2023	Mary Smith	PROF
CS 3140	SPRING 2023	Richard Ngyuen	PROF
APMA 3100	SUMMER 2023	Tammy Ngo	STUD
CS 3140	FALL 2023	William McBurney	PROF
APMA 3100	SUMMER 2023	Yuxi Yang	STUD
CS 3140	SPRING 2023	Yuxi Yang	STUD

Assignment Topic:

```
SELECT
    CRS.Crs_Subject || ' ' || CRS.Crs_CatalogNumber as 'Course',
    S.Sem_Name || ' ' || S.Sem_Year as 'Semester',
    A.Assign_Name as 'Assignment Name',
    group_concat(CT.CrsTopic_Name, ',  ') as 'Assignment Topics'
  FROM AssignmentTopic AT
  JOIN Assignment A USING (Assign_ID)
  JOIN CourseTopic CT USING (CrsTopic_ID)
  JOIN Class CLS USING (Class_ID)
  JOIN Course CRS USING (Crs_ID)
  JOIN Semester S USING (Sem_ID)
  GROUP BY Assign_ID
  ORDER BY "Course", "Semester", "Assignment Name";
```

Table 14
Shows class, assignment name, and related assignment topics

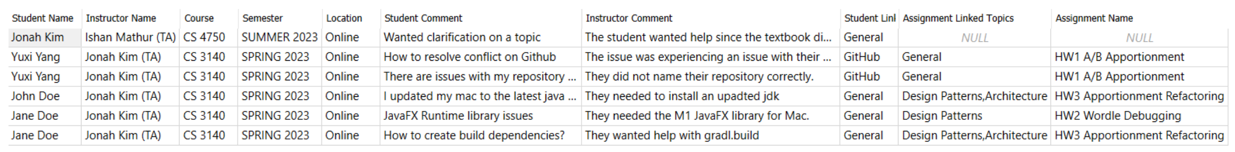


Course	Semester	Assignment Name	Assignment Topics
CS 3140	FALL 2023	HW1 A/B Apportionment	General
CS 3140	FALL 2023	HW1 C Apportionment	Design Patterns, General
CS 3140	FALL 2023	HW2 Wordle Debugging	Design Patterns
CS 3140	FALL 2023	HW3 Apportionment Refactoring	Design Patterns, Architecture
CS 3140	SPRING 2023	HW1 A/B Apportionment	General
CS 3140	SPRING 2023	HW1 C Apportionment	Design Patterns, General
CS 3140	SPRING 2023	HW2 Wordle Debugging	Design Patterns
CS 3140	SPRING 2023	HW3 Apportionment Refactoring	Design Patterns, Architecture
CS 3140	SUMMER 2023	Project Report 1	JDBC
CS 4750	SUMMER 2023	Practice Chapter 6	Normalization
CS 4750	SUMMER 2023	Practice Chapter 7	SQL
CS 4750	SUMMER 2023	Project Final Report	Project, Data Redundancy, Specialization Hierarchies
CS 4750	SUMMER 2023	Project Report 2	Project
CS 4750	SUMMER 2023	Quiz 1	General, Logistics
CS 4750	SUMMER 2023	Quiz 2	General, Logistics

OfficeHourQuestionTopic:

```
SELECT
     Crs_Subject || ' ' || Crs_CatalogNumber as 'Course',
     UM.UnivMem_FirstName || ' ' || UM.UnivMem_LastName as 'Question By',
     group_concat(CrsTopic_Name, ',  ') as 'Linked Topics'
  FROM OfficeHourQuestionTopic
  JOIN OfficeHourQuestion OHQ USING (OHQ_ID)
  JOIN UniversityMember UM USING (UnivMem_ID)
  JOIN CourseTopic CT USING (CrsTopic_ID)
  JOIN Course USING (Crs_ID)
  GROUP BY OHQ_ID
  ORDER BY UM.UnivMem_FirstName, UM.UnivMem_LastName;
```

Table 15
Shows course, the student who asked the question, and the linked topics



Student Name	Instructor Name	Course	Semester	Location	Student Comment	Instructor Comment	Student Linl	Assignment Linked Topics	Assignment Name
Jonah Kim	Ishan Mathur (TA)	CS 4750	SUMMER 2023	Online	Wanted clarification on a topic	The student wanted help since the textbook di...	General	NULL	NULL
Yuxi Yang	Jonah Kim (TA)	CS 3140	SPRING 2023	Online	How to resolve conflict on Github	The issue was experiencing an issue with their ...	GitHub	General	HW1 A/B Apportionment
Yuxi Yang	Jonah Kim (TA)	CS 3140	SPRING 2023	Online	There are issues with my repository ...	They did not name their repository correctly.	GitHub	General	HW1 A/B Apportionment
John Doe	Jonah Kim (TA)	CS 3140	SPRING 2023	Online	I updated my mac to the latest java ...	They needed to install an upadted jdk	General	Design Patterns,Architecture	HW3 Apportionment Refactoring
Jane Doe	Jonah Kim (TA)	CS 3140	SPRING 2023	Online	JavaFX Runtime library issues	They needed the M1 JavaFX library for Mac.	General	Design Patterns	HW2 Wordle Debugging
Jane Doe	Jonah Kim (TA)	CS 3140	SPRING 2023	Online	How to create build dependencies?	They wanted help with gradl.build	General	Design Patterns,Architecture	HW3 Apportionment Refactoring

DJANGO:

Differences in Documentation:

Django supplies its own Object Relational Model which simplifies many of routine database tasks such as defining entities, tables, insertions, updates, etc. However, because of this, there are some inconsistencies in how Django generates its tables when compared to our documentation (such as adding app prefixes). This section outlines some key differences:

Table Names:

Table 16
Shows differences in Django's table generation when compared to the documentation

Documentation's Table Name	Django's Table Name
Assignment	website_assignment
AssignmentTopic	website_assignment_assignment_topics
Class	website_class
Course	website_course
CourseTopic	website_coursetopic
Instructor	website_instructor
Location	website_location
OfficeHourQuestion	website_officehourquestion
OfficeHourQuestionTopic	website_officehourquestion_ohq_topics
OfficeHourSession	website_officehoursession
Professor	website_professor
Registration	website_registration
Semester	website_semester
TeachingAssistant	website_teachingassistant
UniversityMember	website_universitymember

Homonyms and Synonyms:

Within the project, there are conflicts between database design standards and coding standards. Database design standards recommend avoiding the use of homonyms and synonyms whenever possible. This is to improve the quality of the database system and help make the relationships between tables more apparent. However, while coding, one needs variable names which carry semantic value to improve maintainability and analyzability. For example, in OfficeHourQuestion, it is better to have a variable named *student* rather than *university_member*. Django's default behavior when constructing database tables is to pull the attributes from an entity model and use those names as the field names in the database. Consequently, the project employs some homonyms and synonyms within its structure.

Surrogate Keys:

Django's ORM automatically generates surrogate keys for use within models by default. For instance, all ManyToMany relationships are given surrogate keys instead of relying on a composite identifier. Additionally, all surrogate keys are given the placeholder name 'id.' This serves several purposes within Django's framework since most objects have a common identifier to be used in methods; however, this creates differences in the documentation since *UnivMem_ID* and *Reg_ID* both become *id*.

Constraints:

Within the project, the location of some constraints have been moved from the database layer into the business logic/controller layer. The triggers shown in the project documentation cannot be directly represented within Django's models; therefore, these checks have been moved into the view logic.

Project Access:

The project can be accessed one of two ways. Either one can run it locally using Django's development server or one can access it online here. We decided to provide the online version as a backup in case it became too difficult to set up locally. However, our hosting site Heroku does not support SQlite databases so we had to switch to Postgres. Consequently, some of our documentation (system architecture, queries) does not accurately reflect the online project structure.

Project Link: https://office-hour-manager-7006b16d76d2.herokuapp.com/information/home/

Note: **There are issues with time zones within the online app. The heroku server is operating in UTC while some date functions within our app are running in different time zones. Consequently, instructor office hour sessions may not appear when you expect them, and dates/times may appear incorrect. The local project should not have these issues.**

Initial Data:

The project's initial data can be found in the data.json file in the root project directory. This file is loaded into the database by navigating to the 'test/populate' url of the web app; however, the submitted database should already be prepopulated (including additional data not within data.json). The most useful account for testing will be the admin account:

username='admin'
password='123'

This account has access to all webpages. The usernames and passwords can be found in the data.json file. All passwords should be '123' for ease of access.

Local Setup:

An equivalent list of setup instructions should be contained in the project's ReadMe; however, we list them here for ease of reference and with additional details.

1. Clone / unzip project code into a directory
2. Open a terminal in that directory
3. Ensure python3 is installed
4. (*Optional*) Create a python virtual environment: 'python -m venv venv'
    a. Activate:
        i. (Windows cmd.exe): 'venv\Scripts\activate.bat'
        ii. (Windows powershell): 'venv\Scripts\Activate.ps1'
        iii. (Mac/Linux): 'source venv/bin/activate'
5. Install all necessary python libraries with: 'pip install -r requirements.txt'
6. Run: 'python manage.py makemigrations' (should not *need* to be run)
7. Run: 'python manage.py migrate' (should not *need* to be run)
8. Run: 'python manage.py runserver'
    a. You may need to specify the environment variable: DJANGO_SETTINGS_MODULE=manager.settings
    b. During a test setup sequence, it was unnecessary to do so; however, during development this is usually required.
9. Access url from terminal: http://127.0.0.1:8000/