Database Design for an Class Management System

Brandon Ong, Kha Nguyen, Lian Huang

College of Science, San José State University

CS 157A – Introduction to Database Management Systems

Prof. Jahan Ghofraniha

Nov. 9th, 2022

Table of Contents

Executive Summary

The main purpose of this project is to design a database for the class management system. In this project report, we will discuss the background of the existing database management systems and the problems and disadvantages they may have. We propose our database for the class management system which will help with those problems. This report lists all the design step details that are needed for the class database management system, requirement gathering, conceptual design stage, normalization steps for the logic design stage, and physical design stage. It also includes the implementation and testing of the design to check if it works as expected. We will also discuss the front-end and back-end parts for deploying the database design to a real website.

Background/Introduction

The core components of a class are the assigned tasks and the resources to help with their completion. Some examples of tasks are homework, projects, reports, quizzes, midterms, and finals. All of these items have their own differences but they are all centered on a topic or set of topics. Some examples of resources are lectures, textbooks, and notes. All of these are organized based on topics. For example, a database textbook could have a chapter focused on the design process and the professor would give a lecture on that topic.

If a student needs to complete a homework assignment about database normalization, they will refer to their notes on the topic of normalization. If a student needs to study for a quiz, useful resources would include the notes and assignments on the covered topics. From these examples, it is clear that a class's tasks and resources are related to each other based on their

topic. Taking this into consideration, key relationships in our application will be defined based on the topic.

Problem Statement

It is important to maintain order and organization as students, especially when they are overwhelmed with responsibilities and commitments. Without having good time management abilities, students would encounter challenges and problems that would hinder students' success and personal development. Failing to maintain organization would lead students to falling behind deadlines or working at the last minute. As a consequence, their mental state would suffer and deteriorate contributing to symptoms such as stress and anxiety (Simmons, 2018). Not tracking or estimating time needed to complete work is another problem when students do not develop time management. It would threaten their productivity and motivation knowing that they do not use their time effectively causing more time wasted. Therefore, it is crucial to track time and manage them wisely. One of many methods to achieve this is having a to do list. By planning and prioritizing through a note-taking system, everyone could know what to expect. This method makes the tasks more manageable, and it helps students stay focused when they can clearly outline everything. It also reminds students and gives them senses of satisfaction when they accomplish their goals (Stawarz, 2014); as a result, improving students' productivity and performance.

Purpose/Motivation

We are motivated by observing how students are tracking their course assignments, exams, and other course-related resources and notes. The most common way for students to track their assignments, exam, quizzes, or project due date is from the to-do list on Canvas. However, to find the material or notes students took from class or textbook requires going through all the notebook pages or opening the software that students used to save or take notes. To save the look-up time and manage the course-related tasks and material better, we decided to design a database that helps manage their academics more efficiently. The goal of this project is to create a database for all the students that could organize their course tasks and course-related material in one single place and pull up the information they needed fast and simple.

Design

Requirements Gathering

As mentioned earlier, some examples of tasks are homework, projects, reports, quizzes, midterms, and finals. Tasks can be split into two types, assignments and exams. Homework, projects, and reports are just different kinds of assignments so there will be a single assignment entity to cover these items. Quizzes, midterms, and finals are just different kinds of exams, so there will be a single exam entity to cover these items. The initial entities include the following: Class, Notes, Exam, and Assignment.

Students often have multiple assignments and exams to deal with and organizing all of this would greatly help with prioritizing certain tasks. This will be implemented in our app with a view that shows all tasks and their corresponding information like due date,

To do list: assignment id, due date, topic, (related notes)

When a student has an exam they prepare for it by reviewing all related items such as previous assignments, previous exams, and notes. This will be implemented in our app with a study list view that shows all tasks and notes that are relevant to the exam.

Study list: Exam ID, name, due date,(related topics), (related items)

Sometimes students need to refer to information about a specific class. For example, if they need to talk to the professor, they'll need find the information about office hours or how to contact the professor. All the information and details about a class will be put into a class info view.

Class Info: Class ID, course name, meeting time, location, (professor name, email), (office hours time, location)

Conceptual Design

- **Entity and Attributes recognition**

Class: Class_Id, Course_Name, Class_Section, Day, Course_Units, Professor_Name, Professor_Email, Location, Start_Time, End_Time, Office_Hour

Notes: Notes_Id, Topic, Notes_Content, Student_Name

Exam: Exam_Id, Date, Exam_Name, (Topic)

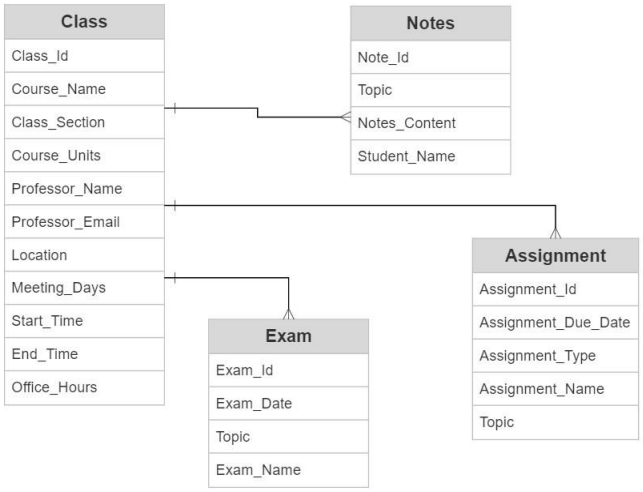
Assignment: Assigment_Id, Assignment_Due_Date, Assignment_type, Assignment_Name, (Topic)

- **Relationship between entities**

A Class can have multiple notes, exams and assignment

One to many : Class to Notes, Class to Exam, Class to Assignment

- **ERD**



Logical Design

- **Normalization process**

1NF: Remove multivalued attributes and repeating groups

In Class entity attributes**(**Course_Name, Course_Units**), (**Professor_Name, Professor_Email, Office_Hour**)** are considered as repeating groups, to make it 1NF we split them into two separate new relations Course and Professor.

In Notes, Assignment and Exam entity attributes (Topic) are considered multivalued, we split them into new relation Topic to satisfy 1NF.

Class: Class_Id(PK), Location, Class_Section, Day, Start_Time, End_Time

Course: Course_Id(PK), Course_Name, Course_Units

Professor: Professor_Id(PK), Professor_Name, Professor_Email, Office_Hour

Notes: Notes_Id(PK), Notes_Content, Student_Name

Exam: Exam_Id(PK), Exam_Date, Exam_Name

Topic: Note_Id(PK), Exam_Id(PK), Topic_Id(PK), Assignment_Id(PK),Topic_Name

Assignment: Assigment_Id(PK), Assignment_Due_Date, Assignment_type, Assignment_Name

2NF: Remove partially dependent attributes

Since a single atomic attribute for its primary key is automatically in second normal form. So 1NF with a single primary key is in 2NF.

For entity Topic, attributes Topic_Name only depend on one of the four primary keys (Topic_Id), we split Topic_Id and Topic_Name as a new relation Topic and change the initial entity name from Topic to Exam_Topics, Assignment_Id split to new relation to Assignment_Topic to satisfy 2NF

Class: Class_Id(PK), Location, Class_Section, Day, Start_Time, End_Time

Course: Course_Id(PK), Course_Name, Course_Units

Professor: Professor_Id(PK), Professor_Name, Professor_Email, Office_Hour

Notes: Notes_Id(PK), Notes_Content, Student_Name

Exam: Exam_Id(PK), Exam_Date, Exam_Name

Topic: Topic_Id(PK), Topic_Name

Assignment_Topic : Topic_Id(PK), Assignment_Id(PK)

Exam_Topics: Note_Id(PK), Exam_Id(PK)

Assignment: Assigment_Id(PK), Assignment_Due_Date, Assignment_type, Assignment_Name

3NF: Remove transitively dependent attributes

In entity Notes, attribute Student_Name is not dependent on Notes_Id, so to make it 3F, we split to a new relation Student

Class: Class_Id(PK), Location, Class_Section, Day, Start_Time, End_Time

Course: Course_Id(PK), Course_Name, Course_Units

Professor: Professor_Id(PK), Professor_Name, Professor_Email, Office_Hour

Notes: Notes_Id(PK), Notes_Content

Student: Student_Id, Student_Name, Student_Password

Exam: Exam_Id(PK), Exam_Date, Exam_Name

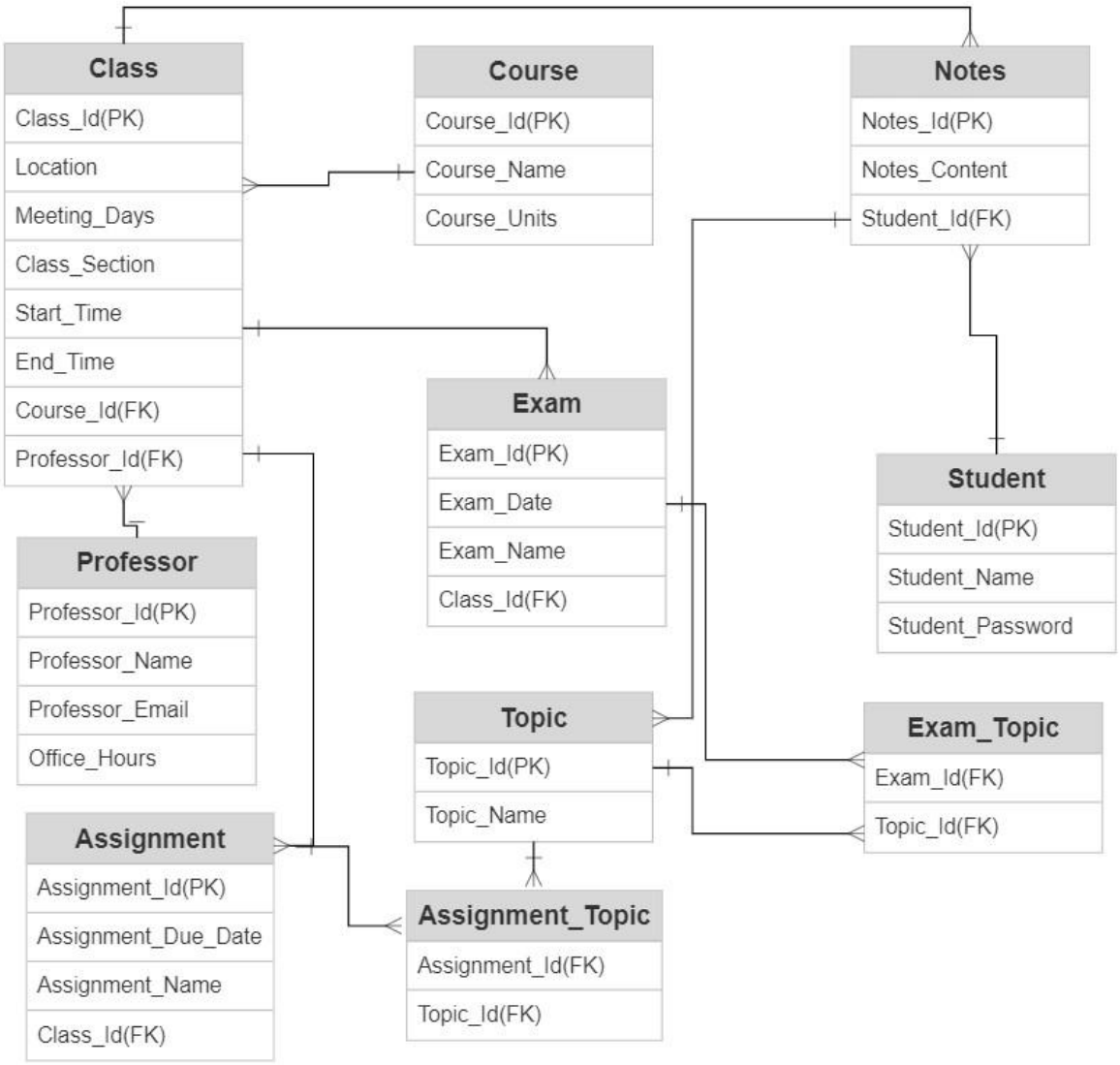
Assignment_Topic : Topic_Id(PK), Assignment_Id(PK)

Topic: Topic_Id(PK), Topic_Name

Exam_Topics: Note_Id(PK), Exam_Id(PK)

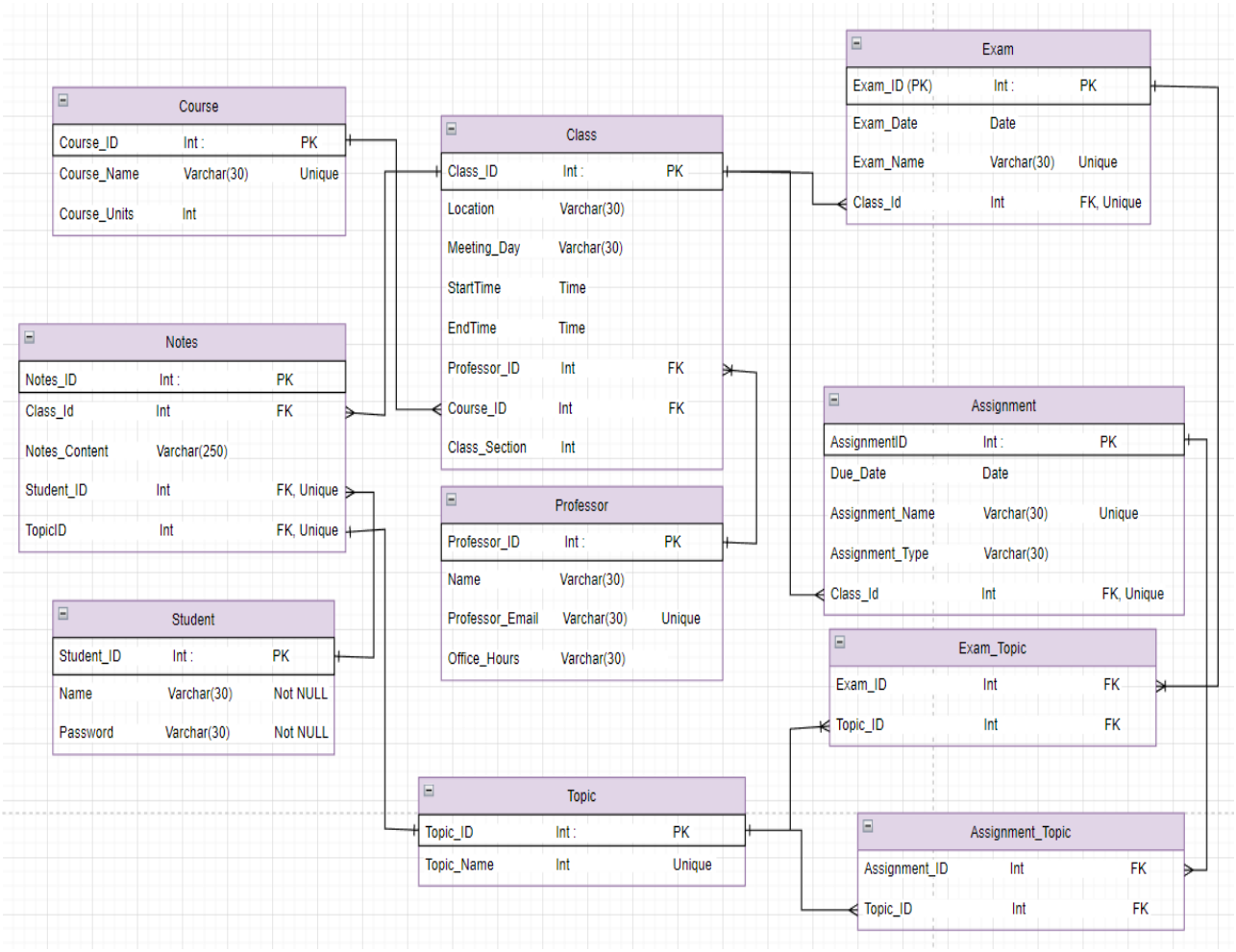
Assignment: Assigment_Id(PK), Assignment_Due_Date, Assignment_type, Assignment_Name

- **ERD**


Class
Class_Id(PK)
Location
Meeting_Days
Class_Section
Start_Time
End_Time
Course_Id(FK)
Professor_Id(FK)
Course
Course_Id(PK)
Course_Name
Course_Units
Notes
Notes_Id(PK)
Notes_Content
Student_Id(FK)
Exam
Exam_Id(PK)
Exam_Date
Exam_Name
Class_Id(FK)
Student
Student_Id(PK)
Student_Name
Student_Password
Professor
Professor_Id(PK)
Professor_Name
Professor_Email
Office_Hours
Topic
Topic_Id(PK)
Topic_Name
Exam_Topic
Exam_Id(FK)
Topic_Id(FK)
Assignment
Assignment_Id(PK)
Assignment_Due_Date
Assignment_Name
Class_Id(FK)
Assignment_Topic
Assignment_Id(FK)
Topic_Id(FK)

Physical Design

- **ERD**



**Partitioning is not required in our design process since our application would not handle a lot of traffic's amount*

Implementation & Testing

SQL create tables

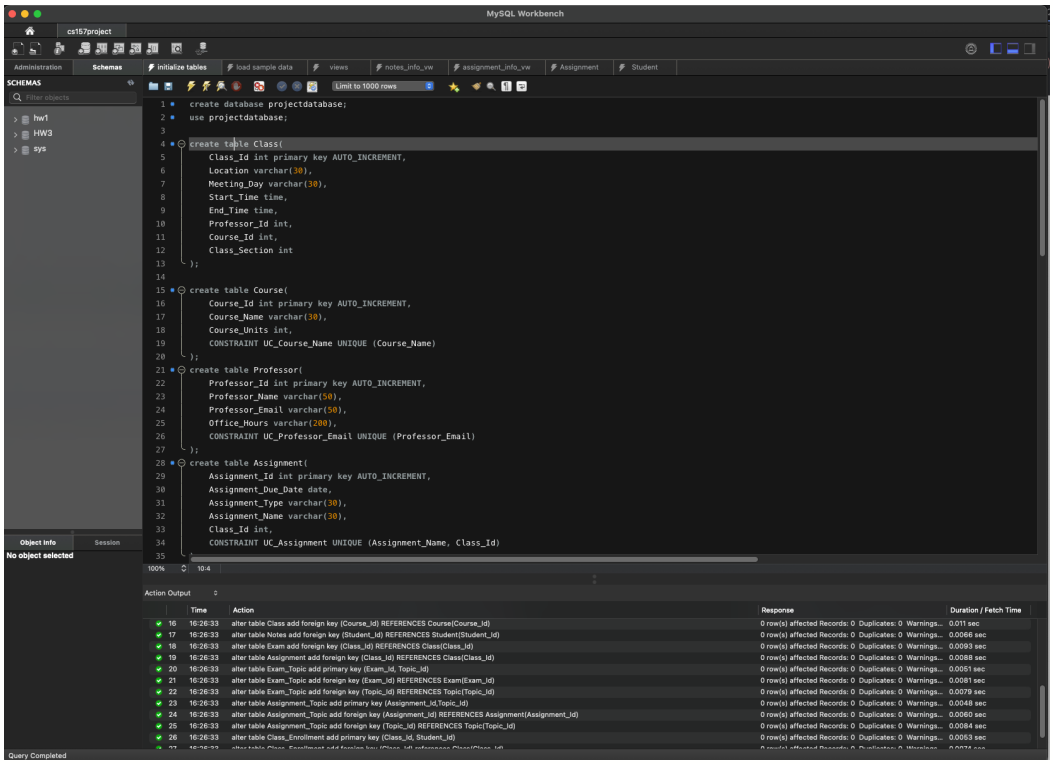
```
create database projectdatabase;

use projectdatabase;


create table Class(
   Class_Id int primary key AUTO_INCREMENT,
   Location varchar(30),
   Meeting_Day varchar(30),
   Start_Time time,
   End_Time time,
   Professor_Id int,
   Course_Id int,
   Class_Section int
);


create table Course(
   Course_Id int primary key AUTO_INCREMENT,
   Course_Name varchar(30),
   Course_Units int,
   CONSTRAINT UC_Course_Name UNIQUE (Course_Name)
);
create table Professor(
   Professor_Id int primary key AUTO_INCREMENT,
   Professor_Name varchar(50),
   Professor_Email varchar(50),
```

```
    Office_Hours varchar(200),
    CONSTRAINT UC_Professor_Email UNIQUE (Professor_Email)
);
create table Assignment(
    Assignment_Id int primary key AUTO_INCREMENT,
    Assignment_Due_Date date,
    Assignment_Type varchar(30),
    Assignment_Name varchar(30),
    Class_Id int,
    CONSTRAINT UC_Assignment UNIQUE (Assignment_Name, Class_Id)
);
create table Student(
    Student_Id int primary key,
    Student_Name varchar(30) NOT NULL,
    Student_Password varchar(30) NOT NULL
);
create table Exam(
    Exam_Id int primary key AUTO_INCREMENT,
    Exam_Date Date,
    Exam_Name varchar(30),
    Class_Id int,
    CONSTRAINT UC_Exam UNIQUE (Exam_Name, Class_Id)
);
create table Notes(
    Notes_Id int primary key AUTO_INCREMENT,
    Notes_Content text,
    Student_Id int,
```

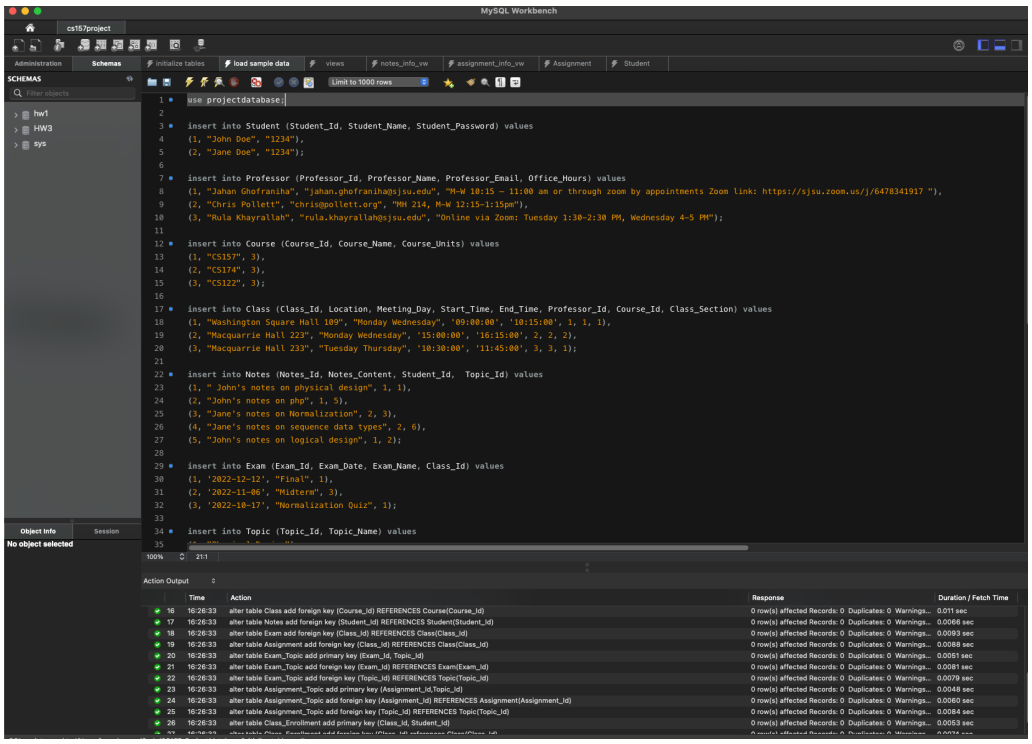
```
    Topic_Id int,
    CONSTRAINT UC_Notes UNIQUE (Student_Id, Topic_Id)
);
create table Topic(
    Topic_Id int primary key AUTO_INCREMENT,
    Topic_Name varchar(30),
    CONSTRAINT UC_Topic_Name UNIQUE (Topic_Name)
);
create table Exam_Topic(
    Exam_Id int,
    Topic_Id int
);
create table Assignment_Topic(
    Assignment_Id int,
    Topic_Id int
);

create table Class_Enrollment(
     Class_Id int,
    Student_Id int
);
alter table Class
add foreign key (Professor_Id) REFERENCES Professor(Professor_Id);
alter table Class
add foreign key (Course_Id) REFERENCES Course(Course_Id);
```

```
alter table Notes
add foreign key (Student_Id) REFERENCES Student(Student_Id);

alter table Exam
add foreign key (Class_Id) REFERENCES Class(Class_Id);

alter table Assignment
add foreign key (Class_Id) REFERENCES Class(Class_Id);

alter table Exam_Topic add primary key (Exam_Id, Topic_Id);
alter table Exam_Topic add foreign key (Exam_Id) REFERENCES Exam(Exam_Id);
alter table Exam_Topic add foreign key (Topic_Id) REFERENCES Topic(Topic_Id);

alter table Assignment_Topic add primary key (Assignment_Id,Topic_Id);
alter table Assignment_Topic add foreign key (Assignment_Id) REFERENCES Assignment(Assignment_Id);
alter table Assignment_Topic add foreign key (Topic_Id) REFERENCES Topic(Topic_Id);

alter table Class_Enrollment add primary key (Class_Id, Student_Id);
alter table Class_Enrollment add foreign key (Class_Id) references Class(Class_Id);
alter table Class_Enrollment add foreign key (Student_Id) references Student(Student_Id);
```

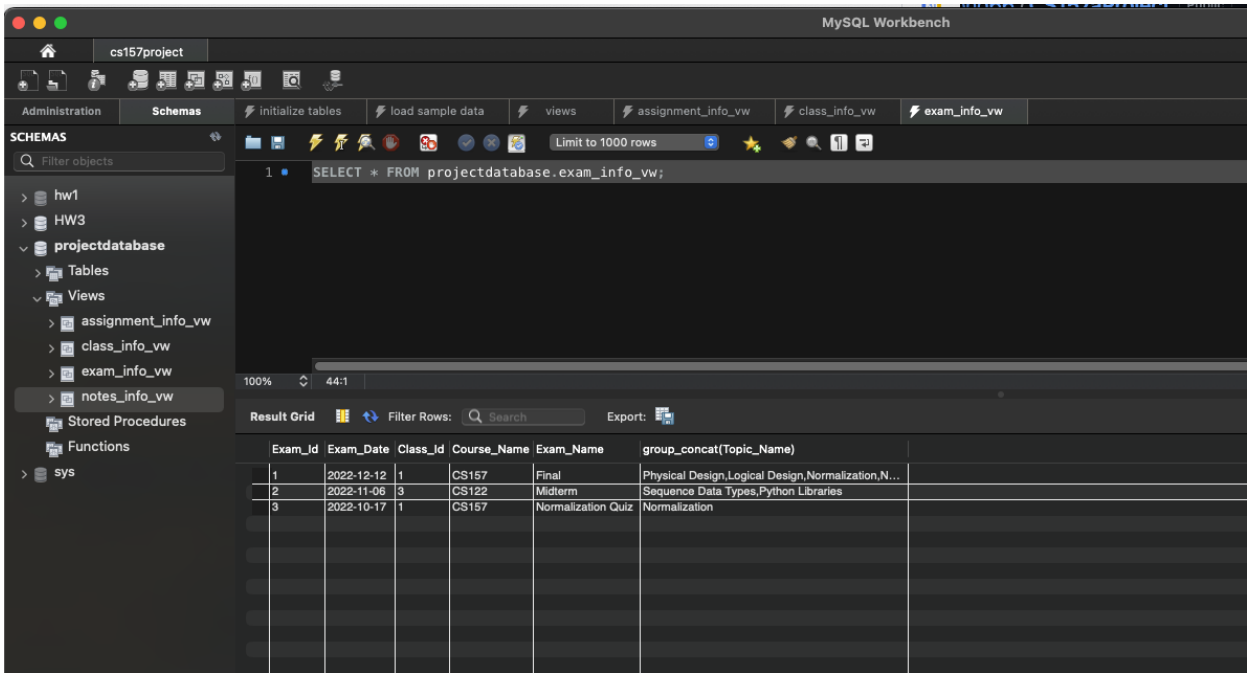
To test our SQL implementation, our technology stack for deployment include MySQL, Node JS, Express JS and React JS



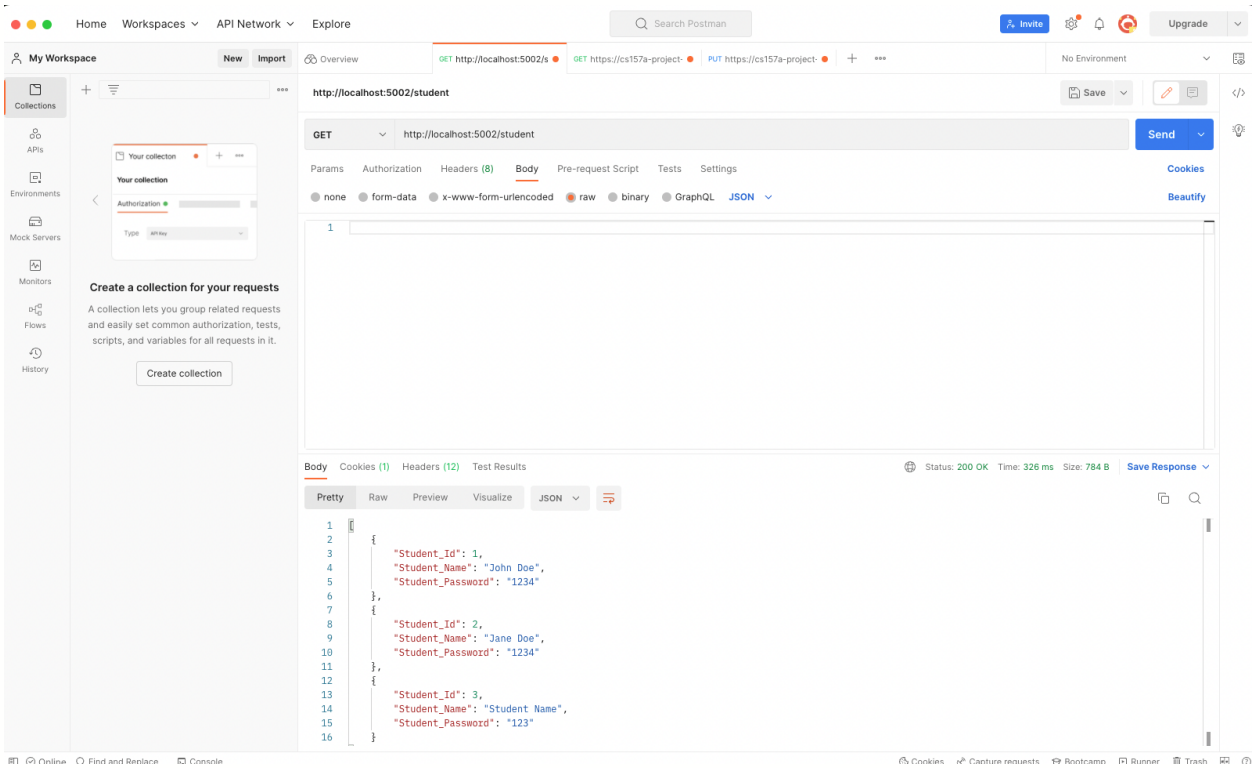
Successfully creating tables in MySQL Workbench



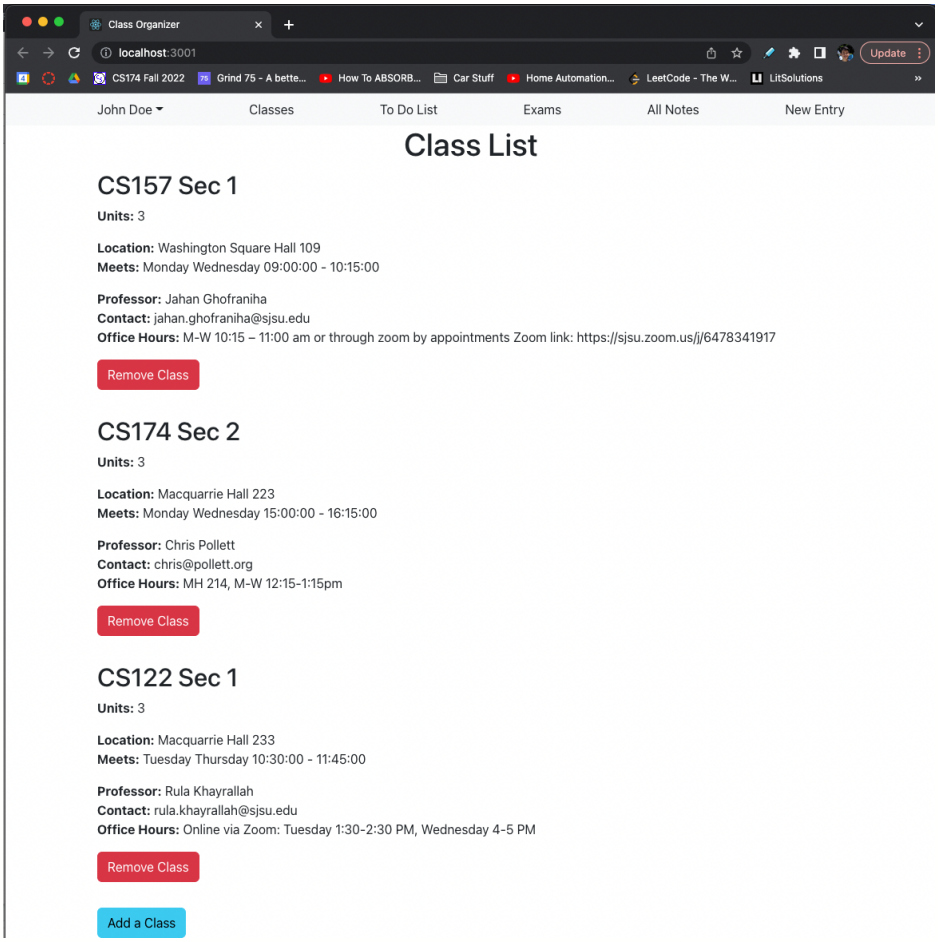
Successfully inserting sample data in MySQL Workbench



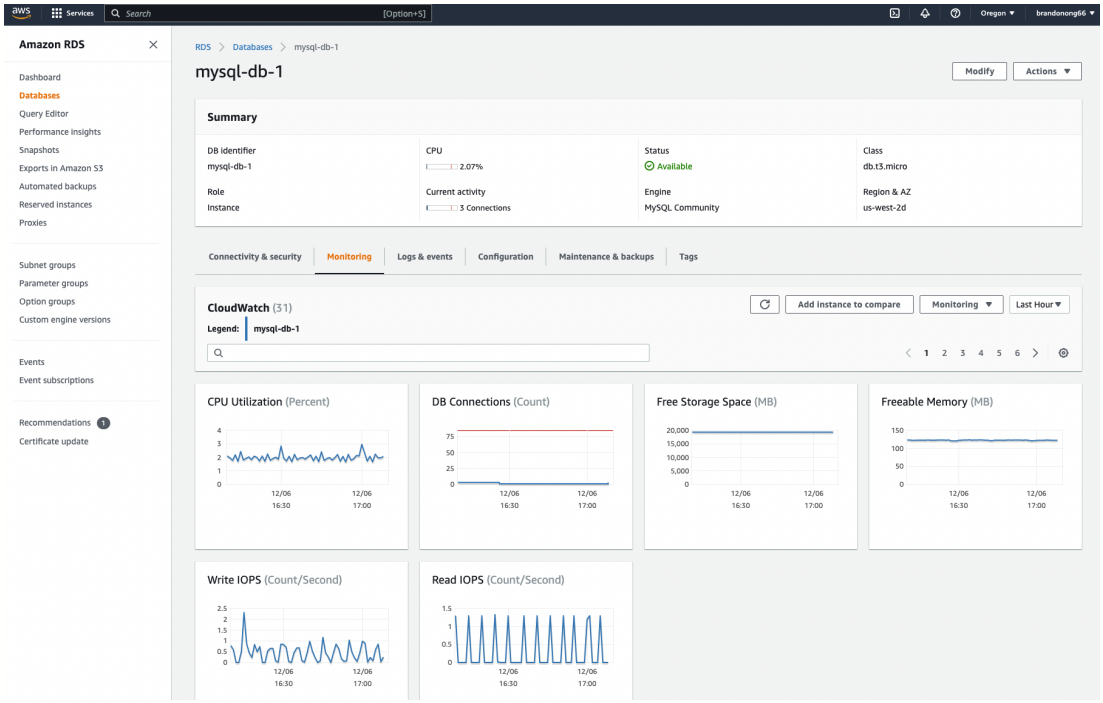
Successfully creating four different Views in MySQL Workbench



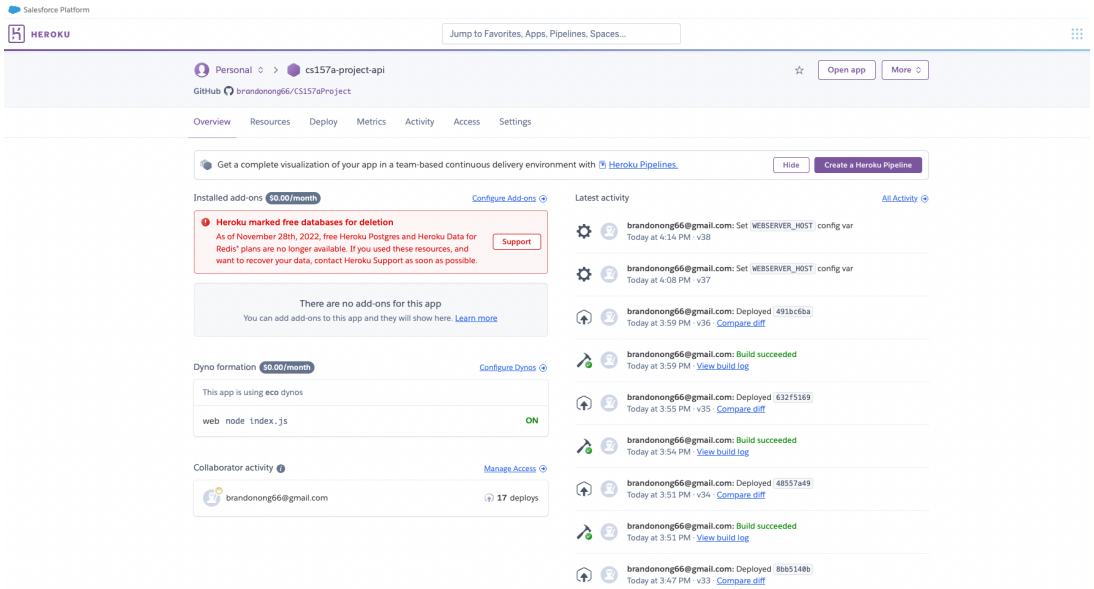
Successfully querying data from Database in API



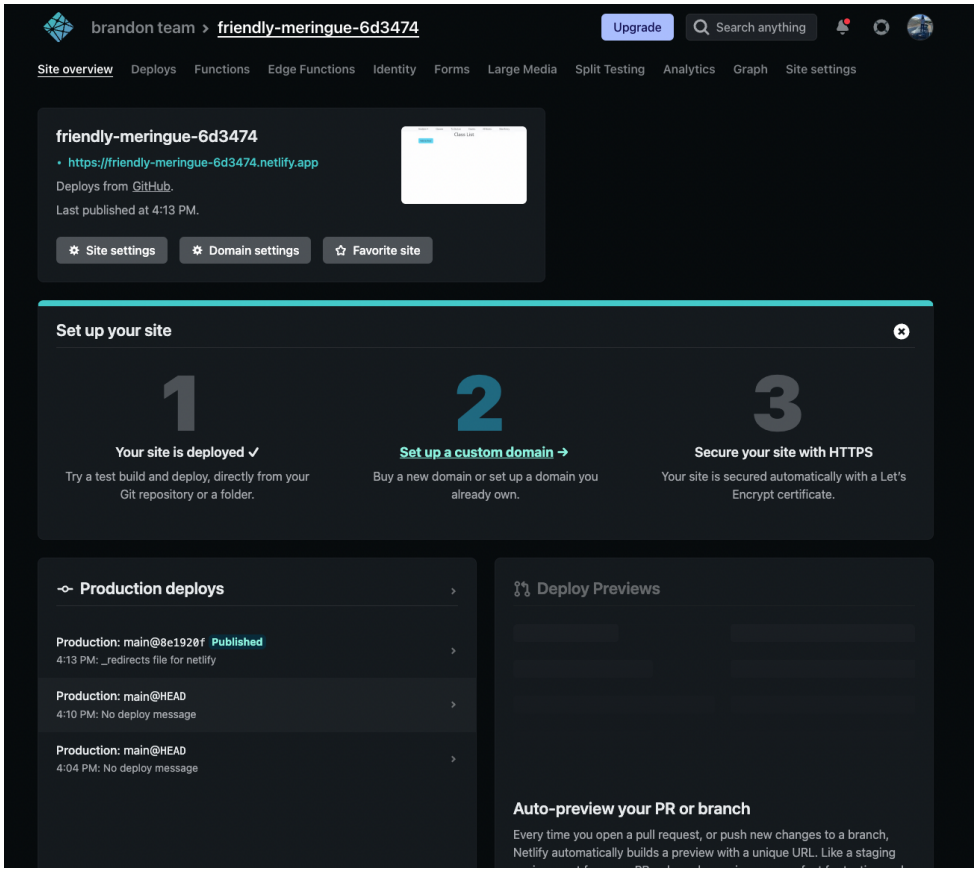
Frontend Implementation successfully shows requesting data



Database hosting on Amazon RDS



Host API with Heroku



Host Frontend with Netlify

Link for deployed website: https://main--friendly-meringue-6d3474.netlify.app/

Conclusion

Through this project, we established and designed a relational database of a note taking and managing application for students. We covered all the required steps to deliver a well-suited database such as requirement gathering, conceptual design, normalization process, logical design, physical design and testing. Our goal is to ensure data integrity to help maintain the consistency and accuracy of storing and retrieving data. Therefore, students can easily access their data and the progress they saved. Furthermore, we learned software tools such as MySQL Workbench and REST API to deploy our database. Additionally, we also create an UI/UX of the application for the client using React to help with easier requests and show different database views. In the end, our implementation and testing results show a smooth communication with the system.

References

Simmons, Crook, A., Cannonier, C., & Simmons, C. (2018). There's an app for that: The impact of reminder apps on student learning and anxiety. Journal of Education for Business, 93(5), 185–195. https://doi.org/10.1080/08832323.2018.1441120

Stawarz, Cox, A., & Blandford, A. (2014). Don't forget your pill: designing effective medication reminder apps that support users' daily routines. Conference on Human Factors in Computing Systems - Proceedings, 2269–2278. https://doi.org/10.1145/2556288.2557079