Scope®

Student's Project
Performance Management Tool

Team #10

Cindy Carrillo

Abraham Tesfay

Yulong Ran

San Jose State University Fall 2019 CS157A

Project Overview

The main purpose of our project is to develop a web application platform that can be used by schools to provide student's performance review based on sets of project requirements the school or instructor's set. Our web application platform called "Scope" will ashier a new opportunity for transparency, improvement, and growth among students.

The Problem

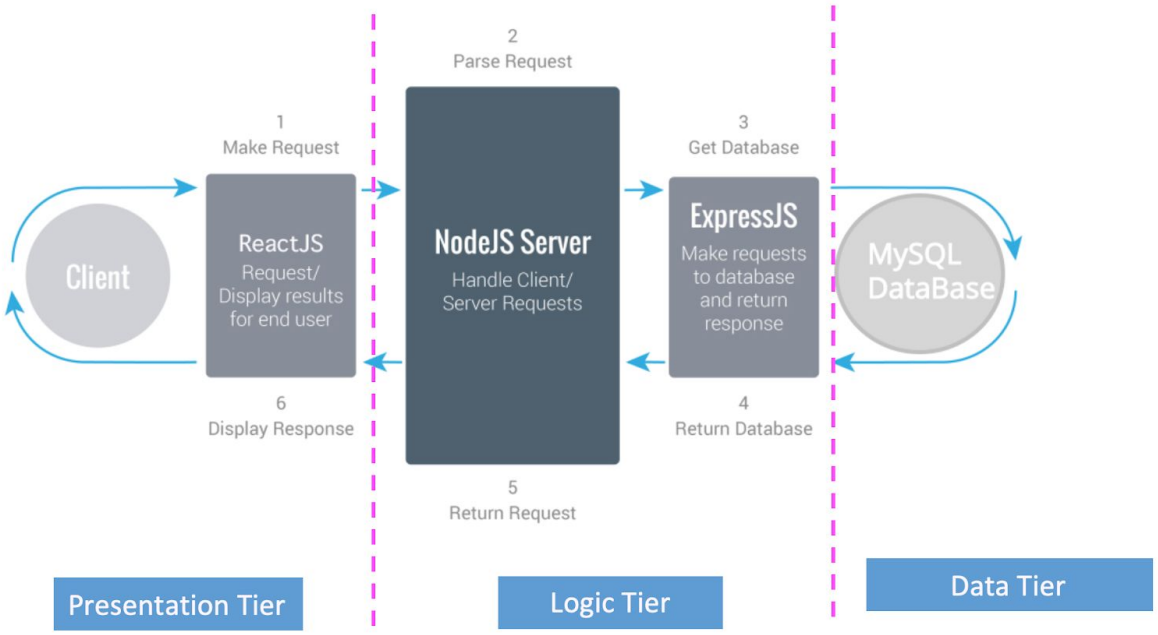
As technology leaps into the next level, the way education is delivered and measured is also rapidly evolving. The traditional way of grading and testing to measure student skills is slowly fading and upgrading with more concrete and skilled based projects that are meant to make students ready for the 'real world'. In real workplace scenario it's a common practice to provide employees with their quarterly work performance review for growth, improvement, and learning opportunity.

At the moment, almost all educational institutes including San Jose State University don't have a tool to track, measure, or provide feedback on student's project performance. Most team projects completed by students are usually tracked and reviewed internally by the professor and this creates a huge gap for transparency, growth and an opportunity for improvement.

Why 'Scope'

Scope will usher a newer, easier, faster and dynamic way to deliver students performance review based on different sets of criteria and milestones that the instructors and the students set. Students will have an opportunity to peer review their team and reflect on their project performance.

System Requirement



Presentation Tier:

In Scope, the user interface will be a web application that handles the interaction with the user and display information. Our team will build this web application with the framework “React” and programming language JavaScript.

Logic Tier:

Scope will use Node.JS as a backend server for handling user requests and communicate with the database.

Data Tier:

Our project will use MySQL as our RDBMS(Relational Database Manage System) to manage and provide access to the data.

Revised Functional Requirements

Users

The users of Scope will be college students who are working on projects for their courses. Users would then have their performance reviewed and can adjust their academic performance, goals, and work habits accordingly. The users would have access to the web application through the university. Instructors can provide links where students can create their one time account that follows them through all their courses at the university that are required to complete their degree.

Functionality and Features

Scope will have a login/sign up page where the student can sign in with university credentials (ID and password) or create an account if they are a new user. Scope will include self assessments which students can complete themselves for their class projects. They can also complete peer performance reviews for students they have worked with and and rate if expectations were met. Student's will have access to view other students' performance reviews to better understand the work habits of those that make up the class.

Inputs/Outputs

The inputs for Scope would include the entered student data such as name, login information, and reviews. The projected outputs would be the ability to view and collect performance reviews for the users from the database and display on the user interface.

Functional Requirements

Display course:

- When a user clicks the Dashboard menu on the navigation bar, a dashboard containing a list of courses will appear on the application.
- The system shall retrieve course data from the database and display a list of courses.

Display Project details:

- When a user clicks the course in the Dashboard, the Dashboard shall refresh its content and display a list of the projects in the course with its name, description, and number of teams.

- The system shall retrieve the project name, description, and number of teams in the database for each project and display on the dashboard as an individual component.

Add Project:

- The user has the ability to create a project by clicking the add project button on the top right of the dashboard. After clicking the add button, the web will have a pop-up form that asks for the name, description of the project, and the number of teams in the project.
- The system shall update the database with a new project under the course.

Delete Project:

- The user has the ability to delete a project by clicking the delete button on the bottom of the project card in the dashboard.
- The system shall update the database and delete the project information.

Add Review:

- The user is able to review by clicking the review button on the top right corner for each project in the project page. After clicking the “review” button, a new page will pop-up, and the user can view reviews from team members or create reviews for team members. The user can view the list of milestones, description, and feedback.
  Example:
  Milestone #1: Project Proposal [Unsatisfactory, Improvement needed, Meets expectations, Exceeds expectations]

Display Team Information:

- When a user clicks a project component in the Dashboard, the Dashboard shall display a list of teams who participated in the project with their team names, description, and number of members.
- The system shall retrieve a list of teams and their basic team information within the course from the database. The user interface shall refresh the dashboard to display updated information.

Display Team members:

- When a user clicks the team number for a specific project, the Dashboard shall refresh its content and display a list of the students in the team with their name.
- The system shall retrieve the student name and id in the database for each student and display on the dashboard as an individual component.

Add / Edit Team:

- A user has the ability to edit a team by clicking on the team and add a team when creating a project.
- The system shall update the database based on the user action.

Self Assessment:

- A student user shall be able to review themselves for each project they are assigned to. The student user will navigate to the project tab in the navigation bar, select which team, then click the review button and specify which milestone to self evaluate.
- The system shall retrieve the student self assessment in the database and display only for the student user to see and reflect.

Sign-in:

- A user has the ability to sign in to Scope using email address and password credentials.
- The system shall update the database based on the user's information.

Sign-up:

- A user has the ability to sign up for a Scope account using name, email address, and password credentials.
- The system shall update the database based on the user's information.

Non Functional Requirements

Non-functional requirements include many aspects such as performance, scalability, accessibility, security, etc. Within the time of CS157A course, our team aims to build a three-tier architecture application that provides student's performance review based on their projects within the range of single University.

In terms of scalability, students' performance review will follow them from course to course and school to school. Student's from different Universities will be able to review each others project performance history.

For accessibility, Scope aims to bring high-performance websites for our users. Users can be able to access our site either on their PC or mobile platform. The navigation system for users will be simple, intuitive, and provide enough information that matches the user's needs.

For security, Scope is designed for university students only and will be synced with each student's university credentials. Students will have the university or instructor assign the courses they have access to based off enrollment.

For GUI mockup and prototype designing, we will be using Adobe's Photoshop and Sketch design, and prototyping tools.

Usability

- User will be able to understand easily the way the program application function.
- User should be able to create project, assign project objectives and review the student's project performance.
- System administrators will be able to use the database to effectively manage projects, student accounts and user information without any error or downtime.
- The application usability is designed to be easily accessed by students.

Reliability

- System will see minimal downtime apart from maintenance and errors.

- Users will not encounter system failures, exceptions will be caught, and users will be given useful information to avoid any error that might arise.
- User's personal information will be safe, secure, and encrypted.
- There will be easy steps for troubleshooting.

Performance

- The system should be able to support larger number of users without affecting any performance of the system operations.
- The system should be able to support a large number of projects and user database transaction simultaneously.
- The database should be able to show the correct project tasks and other related data that specifically assigned the corresponding instructors and students
- Users shall be able to find relevant projects and students by searching through via the Scope platform.
- Loading the database to the system should take no more than a few seconds per page given the user's machine required the minimal device requirements.

Supportability

- The system should allow to fix bugs for any competent.
- The system should allow to add additional functions to adapt changes.
- The system should work on any operating systems.
- The system should work on any size of screen.
- For development the system should work on React JS, Node JS, and MySQL runtime environment implemented on locally or on remote cloud server

Implementation

- This system will implement the Oracle's MySQL database, allowing the user to perform a variety of database transactions related to the project performance tool.
- This system will be written using React JS, MySQL query and JavaScript.
- This system will be hosted by an online web host servicer and also be available in local machine

- This system implements adding, editing, updating and deleting of users, related projects, projects objectives, and milestones.
- The system implements the functionality of producing reports and invoices to keep track of productivity and performance of projects and the stakeholders who are involved.
- The system will save user's activities, profiles, and their tasks in a timely manner that suit the project requirements
- The system will also keep track of the status of projects, tasks, and milestones and notify users appropriately.
- The system will also allow administrators of the system to add new users, manage the account and set different privileges.
- The project will show or produce overall report of the status of the projects, timelines and milestones to show overall performance of a class project.

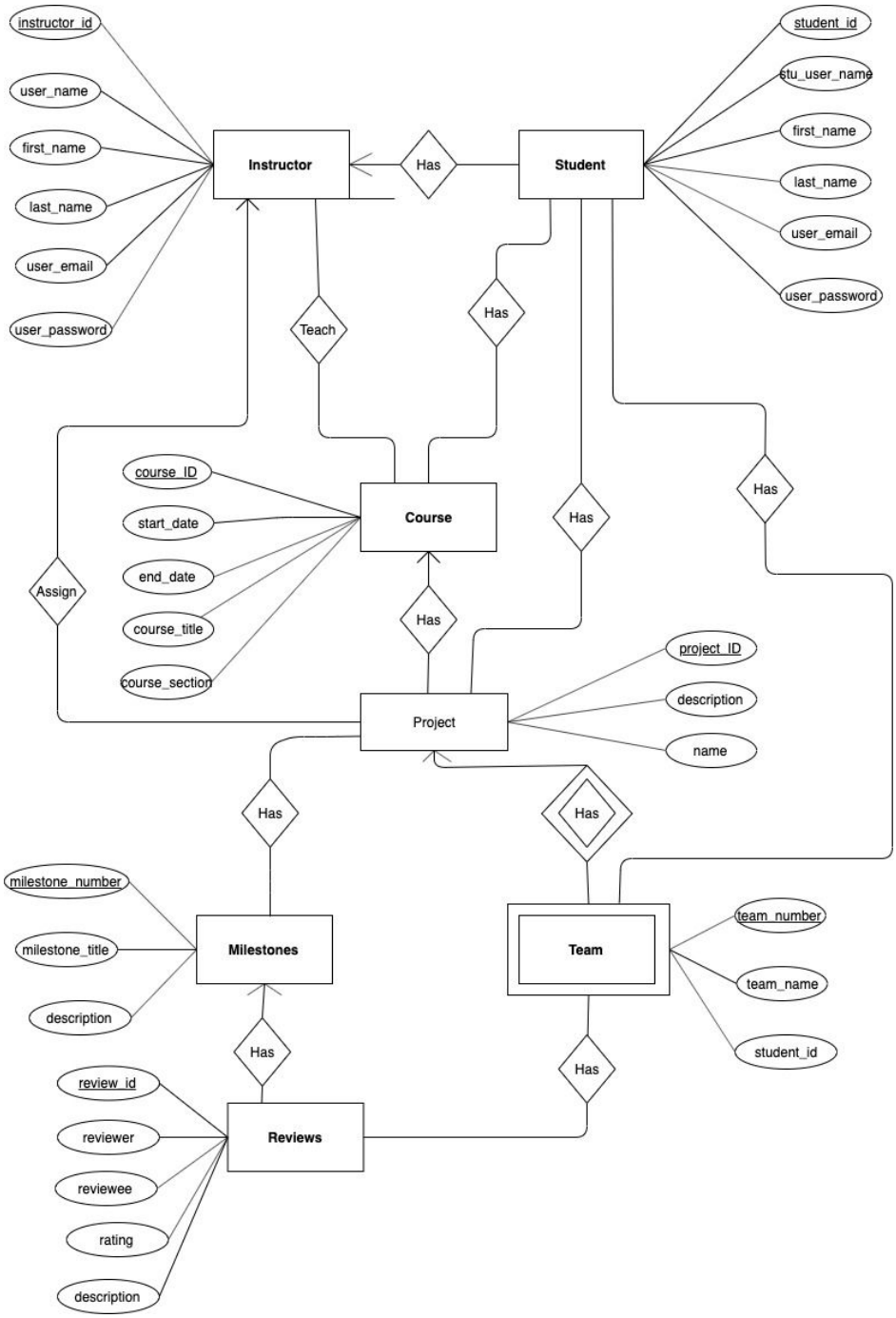
Interface

- This system will interface via online web application using a remote hosting service or can be implemented using local machine running Node Js web server.
- All user requests will be handled in a web browser.

Legal

- This project will not be legally licensed under any professional contracts.
- The project does not display company information, business names, place of registration, registered number, and registered office address. It is only intended for the sole purpose of a school project.
- The system will display a Privacy Policy informing the user what the business does with the data and that it conforms to the The Data Protection Act.
- The Privacy Policy needs to explain what cookies the website will create and what they are for. (Only for the sole purpose of the school project)
- Have Terms & Conditions, user Data Privacy Policy pages to show information on how user data is collected

E/R Diagram


instructor_id
user_name
first_name
last_name
user_email
user_password
Instructor
Has
Student
student_id
stu_user_name
first_name
last_name
user_email
user_password
Teach
Has
Has
course_ID
start_date
end_date
course_title
course_section
Course
Has
Assign
Has
project_ID
description
name
Project
Has
Has
milestone_number
milestone_title
description
Milestones
Team
team_number
team_name
student_id
Has
Has
review_id
reviewer
reviewee
rating
description
Reviews

Description

User

User is an entity that represents the user of Scope. A user must have user_id, user_name, first_name, last_name, user_email, and user_password. The user_id is the primary key for the User table.

Instructor

Instructor is a subclass of the Users entity. The attribute type_instructor has a boolean data type that indicates the identity of the user.

Student

Student is a subclass of the Users entity. The attribute type_student has a boolean data type that indicates the identity of the user.

StudentHasCourse

StudentHasCourse is a relationship that connects Student and Course entity sets. Course can have many students and student can be enrolled in many courses.

StudentHasTeam

StudentHasTeam is a relationship that connects Student and Team entity sets. Teams can have many students and students can belong to many teams.

InstructorHasStudent

InstructorHasStudent is a relationship that indicates the instructor has many students, but students can only have exactly one instructor.

InstructorTeachCourse

InstructorTeachCourse is a relationship that indicates the instructor teaches many courses, but each course can only have exactly one instructor.

InstructorAssignProject

InstructorAssignProject is a relationship that connects Instructor and Project entity sets. Instructor can assign zero to many projects. Project can assign by only one instructor.

Course

Course is an entity that represents the course that the student is in and the instructor teaches. A course must have start_time, end_time, course_id, and course_title. The course_id is the primary key for the table Course.

Project

Project is an entity set that represents the project assigned by the instructor for the student to have. A Project must have attributes of project_id, description, and name. It also has a relation with the Course, Milestone and Team set of entities.

CourseHasProject

CourseHasProject is a relationship that connects the Course and Project entity sets. Each Project can only be in one Course.

Milestone

The Milestone is an entity set that represents the milestones of each Project entity set. A Milestone must have milstone_number, title, description, reviewer, reviewee, and rating.

ProjectHasMilestone

ProjectHasMilestone is a relationship that connects the Project and Milestone entity sets. A single project can have multiple Milestones.

Team

Team is an entity set that represents the Team. Teams are created and assigned. Each Team entity set should contain team_number, team_memebr, and team_name of attributes.

TeamHasReviews

TeamHasReview is a relationship that connects Team and Reviews entity sets. Every Team can have one or more set of reviews from the team member.

TeamHasProject

TeamHasProject is a relationship that connects Team and Project entity sets. Every Team can have one or more sets of projects.

Reviews

Reviews is an entity set that represents the Review table. Every Review is also connected with Milestone entity set and every milestone at least have one review. Reviews entity set must have reviewer, reviewee, and rating set of attributes.

MilestoneHasReview

MilestoneHasReviews is a set of relationships that connects Milestone and Review entity sets. Every Milestone have at least one Review from a students.

Scope Database Schema

Database Name: Scope

Database : Entity Set

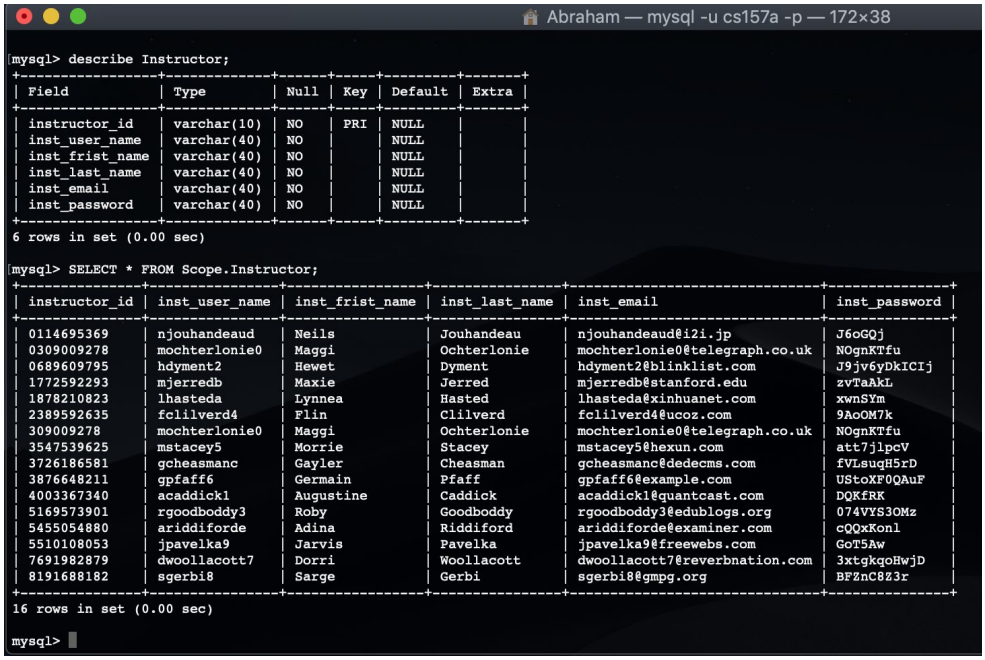
1. Instructor (instructor_id, inst_user_name, inst_frist_name, inst_last_name, inst_email, inst_password)
2. Student (student_id, stu_user_name, stu_frist_name, stu_last_name, stu_email, student_password)
3. Course (course_id, course_name, course_section, start_date, end_date)
4. Project(project_id, project_description , project_name)
5. Milestones(milestone_number, milestone_title, milestone_description, review_id)
6. Team(team_number, team_name, student_id)
7. Reviews(review_id, reviewer, reviewee, rating)

Database : Relationship

1. InstructorHasStudents(instructor_id, student_id)
2. InstructorTeachCourses(instructor_id, course_id)
3. StudentHasCourses(student_id, course_id)
4. InstructorAssignProjects(instructor_id, project_id)
5. StudentHasProjects(student_id, project_id)
6. StudentHasTeams(student_id, project_id, team_number)
7. ProjectHasMilestones(project_id, milestone_number)
8. MilestoneHasReviews (milestone_number, review_id)
9. ProjectHasTeams(project_id, project_id, team_number)
10. TeamHasReviews(project_id, team_number, review_id)

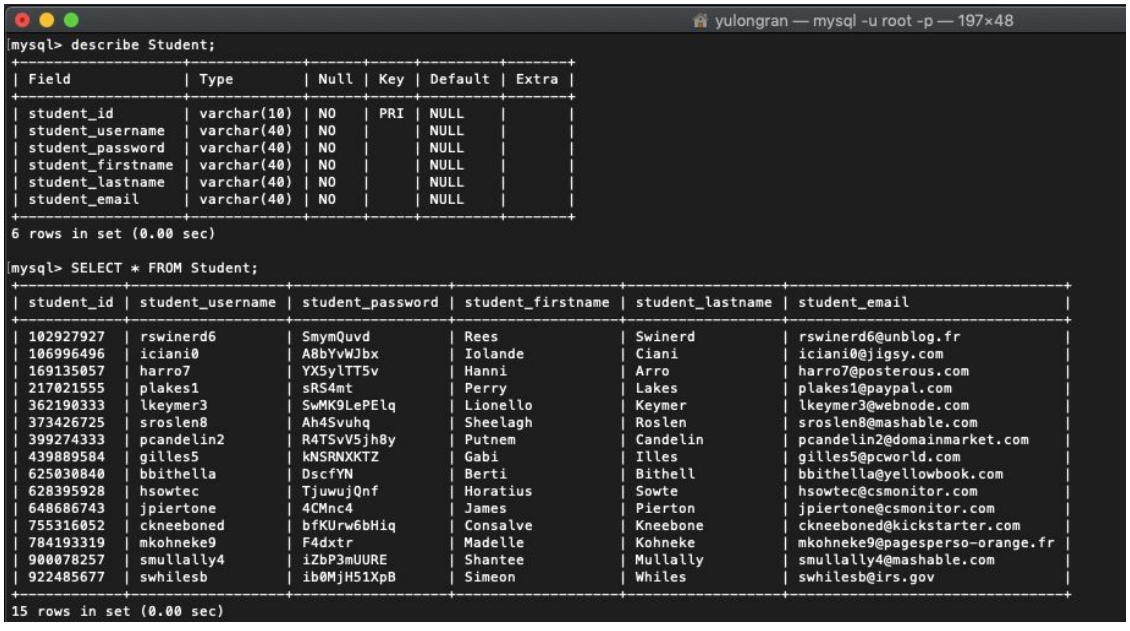
Tables

Instructor Table




```
mysql> describe Instructor;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| instructor_id   | varchar(10) | NO   | PRI | NULL    |       |
| inst_user_name  | varchar(40) | NO   |     | NULL    |       |
| inst_frist_name | varchar(40) | NO   |     | NULL    |       |
| inst_last_name  | varchar(40) | NO   |     | NULL    |       |
| inst_email      | varchar(40) | NO   |     | NULL    |       |
| inst_password   | varchar(40) | NO   |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM Scope.Instructor;
+---------------+----------------+-----------------+----------------+-----------------------------+---------------+
| instructor_id | inst_user_name | inst_frist_name | inst_last_name | inst_email                  | inst_password |
+---------------+----------------+-----------------+----------------+-----------------------------+---------------+
| 0114695369    | njouhandeaud   | Neils           | Jouhandeau     | njouhandeaud@i2i.jp         | J6oGQj        |
| 0309009278    | mochterlonie0  | Maggi           | Ochterlonie    | mochterlonie0@telegraph.co.uk | NOgnKTfu    |
| 0689609795    | hdyment2       | Hewet           | Dyment         | hdyment2@blinklist.com      | J9jv6yDkICIj  |
| 1772592293    | mjerredb       | Maxie           | Jerred         | mjerredb@stanford.edu       | zvTaAkL       |
| 1878210823    | lhasteda       | Lynnea          | Hasted         | lhasteda@xinhuanet.com      | xwnSYm        |
| 2389592635    | fclilverd4     | Flin            | Clilverd       | fclilverd4@ucoz.com         | 9AoOM7k       |
| 309009278     | mochterlonie0  | Maggi           | Ochterlonie    | mochterlonie0@telegraph.co.uk | NOgnKTfu    |
| 3547539625    | mstacey5       | Morrie          | Stacey         | mstacey5@hexun.com          | att7jlpcV     |
| 3726186581    | gcheasmanc     | Gayler          | Cheasman       | gcheasmanc@dedecms.com      | fVLsuqH5rD    |
| 3876648211    | gpfaff6        | Germain         | Pfaff          | gpfaff6@example.com         | UStoXF0QAuF   |
| 4003367340    | acaddick1      | Augustine       | Caddick        | acaddick1@quantcast.com     | DQKfRK        |
| 5169573901    | rgoodboddy3    | Roby            | Goodboddy      | rgoodboddy3@edublogs.org    | 074VYS3OMz    |
| 5455054880    | ariddiforde    | Adina           | Riddiford      | ariddiforde@examiner.com    | cQQxKonl      |
| 5510108053    | jpavelka9      | Jarvis          | Pavelka        | jpavelka9@freewebs.com      | GoT5Aw        |
| 7691982879    | dwoollacott7   | Dorri           | Woollacott     | dwoollacott7@reverbnation.com | 3xtgkqoHwjD |
| 8191688182    | sgerbi8        | Sarge           | Gerbi          | sgerbi8@gmpg.org            | BFZnC8Z3r     |
+---------------+----------------+-----------------+----------------+-----------------------------+---------------+
16 rows in set (0.00 sec)

mysql>
```

Student Table




```
mysql> describe Student;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| student_id       | varchar(10) | NO   | PRI | NULL    |       |
| student_username | varchar(40) | NO   |     | NULL    |       |
| student_password | varchar(40) | NO   |     | NULL    |       |
| student_firstname| varchar(40) | NO   |     | NULL    |       |
| student_lastname | varchar(40) | NO   |     | NULL    |       |
| student_email    | varchar(40) | NO   |     | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM Student;
+------------+------------------+------------------+------------------+------------------+-------------------------------+
| student_id | student_username | student_password | student_firstname | student_lastname | student_email                 |
+------------+------------------+------------------+------------------+------------------+-------------------------------+
| 102927927  | rswinerd6        | SmymQuvd         | Rees             | Swinerd          | rswinerd6@unblog.fr           |
| 106996496  | iciani0          | A8bYvWJbx        | Iolande          | Ciani            | iciani0@jigsy.com             |
| 169135057  | harro7           | YX5ylTT5v        | Hanni            | Arro             | harro7@posterous.com          |
| 217021555  | plakes1          | sRS4mt           | Perry            | Lakes            | plakes1@paypal.com            |
| 362190333  | lkeymer3         | SwMK9LePElq      | Lionello         | Keymer           | lkeymer3@webnode.com          |
| 373426725  | sroslen8         | Ah4Svuhq         | Sheelagh         | Roslen           | sroslen8@mashable.com         |
| 399274333  | pcandelin2       | R4TSvV5jh8y      | Putnem           | Candelin         | pcandelin2@domainmarket.com   |
| 439889584  | gilles5          | kNSRNXKTZ        | Gabi             | Illes            | gilles5@pcworld.com           |
| 625030840  | bbithella        | DscfYN           | Berti            | Bithell          | bbithella@yellowbook.com      |
| 628395928  | hsowtec          | TjuwujQnf        | Horatius         | Sowte            | hsowtec@csmonitor.com         |
| 648686743  | jpiertone        | 4CMnc4           | James            | Pierton          | jpiertone@csmonitor.com       |
| 755316052  | ckneeboned       | bfKUrw6bHiq      | Consalve         | Kneebone         | ckneeboned@kickstarter.com    |
| 784193319  | mkohneke9        | F4dxtr           | Madelle          | Kohneke          | mkohneke9@pagesperso-orange.fr |
| 900078257  | smullally4       | iZbP3mUURE       | Shantee          | Mullally         | smullally4@mashable.com       |
| 922485677  | swhilesb         | ib0MjH51XpB      | Simeon           | Whiles           | swhilesb@irs.gov              |
+------------+------------------+------------------+------------------+------------------+-------------------------------+
15 rows in set (0.00 sec)
```

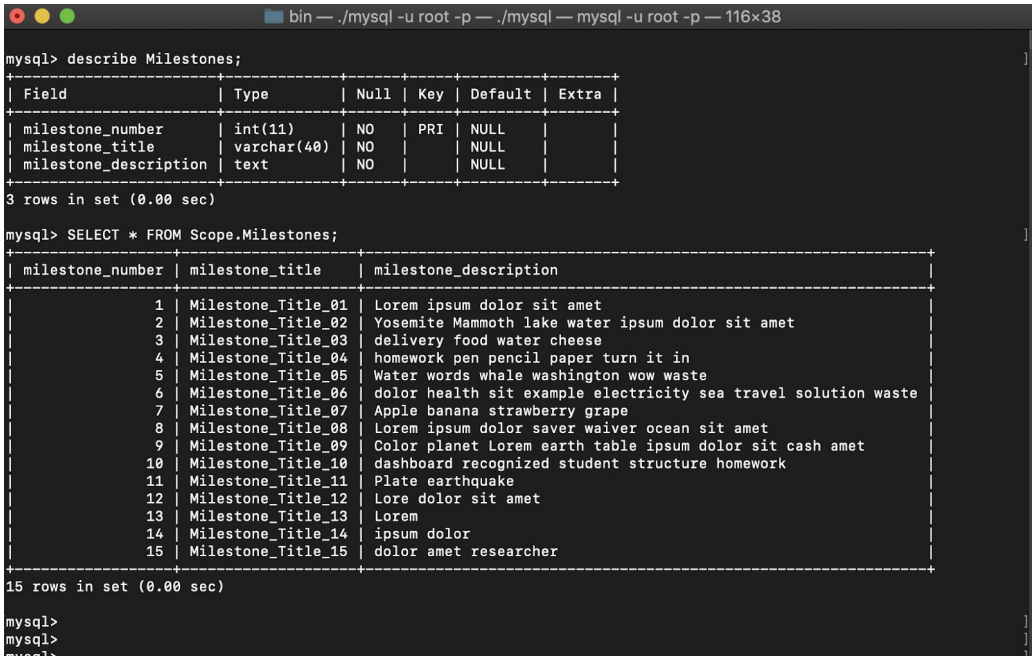
Course Table

```
mysql> describe Course;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| course_id      | varchar(10) | NO   | PRI | NULL    |       |
| course_name    | varchar(40) | NO   |     | NULL    |       |
| course_section | varchar(2)  | NO   |     | NULL    |       |
| start_date     | date        | NO   |     | NULL    |       |
| end_date       | date        | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM Course;
+-----------+-------------+----------------+------------+------------+
| course_id | course_name | course_section | start_date | end_date   |
+-----------+-------------+----------------+------------+------------+
| 278917254 | CS134       | 01             | 2018-08-21 | 2019-12-09 |
| 299655012 | CS166       | 01             | 2018-08-21 | 2019-12-09 |
| 426383441 | CS159       | 01             | 2018-08-21 | 2019-12-09 |
| 55116348  | CS149       | 01             | 2018-01-24 | 2019-05-13 |
| 562182226 | CMPE172     | 01             | 2018-01-24 | 2019-05-13 |
| 567291101 | CS157A      | 01             | 2013-08-21 | 2014-12-09 |
| 593726155 | CS152       | 01             | 2014-01-24 | 2015-05-13 |
| 624040817 | CMPE131     | 01             | 2018-01-24 | 2019-05-13 |
| 629866888 | CS146       | 01             | 2018-08-21 | 2019-12-09 |
| 760306549 | CMPE165     | 02             | 2017-08-21 | 2018-12-09 |
| 868029873 | ENGR195A    | 01             | 2018-01-24 | 2019-05-13 |
| 878839432 | CMPE102     | 01             | 2018-08-21 | 2019-12-09 |
| 927885960 | CMPE165     | 01             | 2015-08-21 | 2016-12-09 |
| 996841846 | ISE164      | 01             | 2018-08-21 | 2019-12-09 |
| 997503965 | CMPE187     | 01             | 2018-01-24 | 2019-05-13 |
+-----------+-------------+----------------+------------+------------+
15 rows in set (0.00 sec)
```

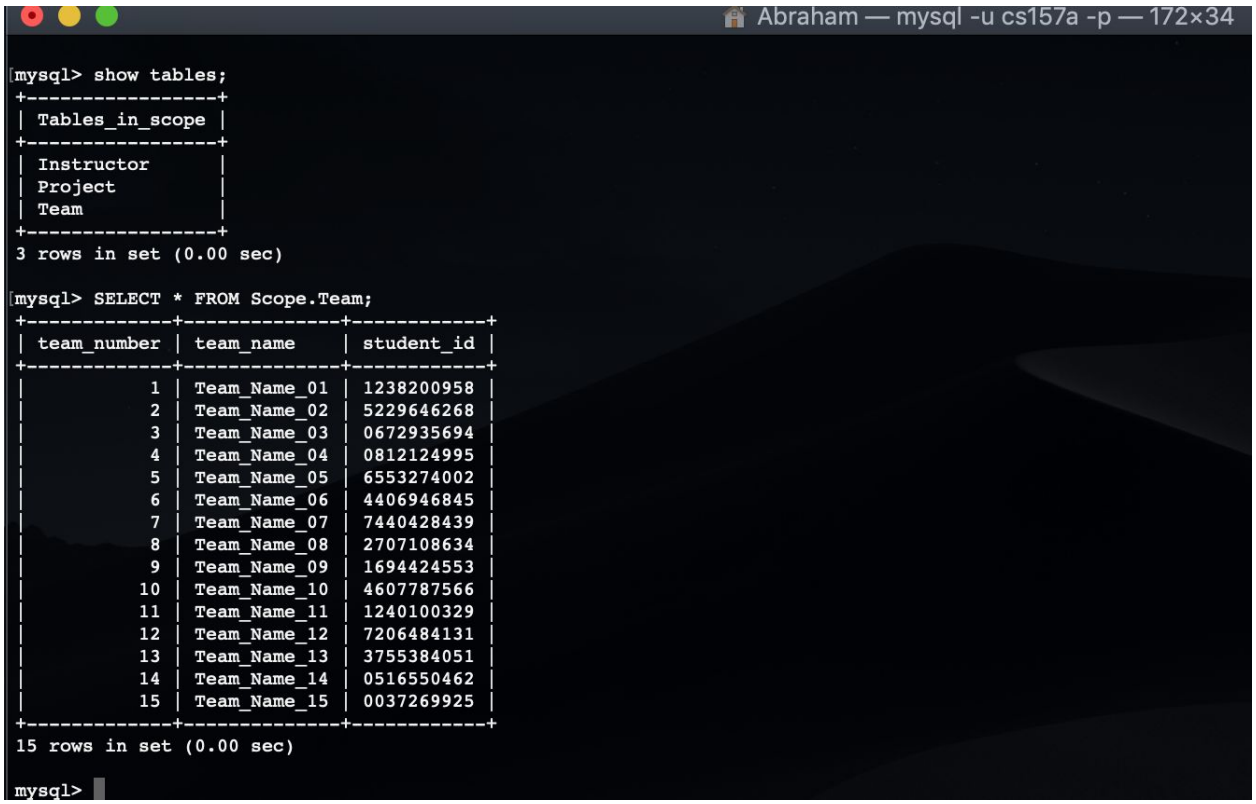
Project Table

```
mysql> show tables;
+-----------------+
| Tables_in_scope |
+-----------------+
| Instructor      |
| Project         |
+-----------------+
2 rows in set (0.01 sec)

mysql> SELECT * FROM Scope.Project;
+------------+--------------+------------------------------------------------------------------------------------------------------------+
| project_id | project_name | project_description                                                                                        |
+------------+--------------+------------------------------------------------------------------------------------------------------------+
|          1 | Title 1      | Laceration with foreign body of unspecified front wall of thorax without penetration into thoracic cavity, subsequent encounter |
|          2 | Title 2      | Poisoning by unspecified antiepileptic and sedative-hypnotic drugs, intentional self-harm, sequela        |
|          3 | Title 3      | Adhesions due to foreign body accidentally left in body following other procedure, initial encounter       |
|          4 | Title 4      | Other specified injury of unspecified blood vessel at ankle and foot level, unspecified leg                |
|          5 | Title 5      | Pedestrian on foot injured in collision with two- or three-wheeled motor vehicle in traffic accident, subsequent encounter |
|          6 | Title 6      | Malnutrition in the puerperium                                                                             |
|          7 | Title 7      | Toxic effect of arsenic and its compounds, accidental (unintentional), initial encounter                  |
|          8 | Title 8      | Osteonecrosis due to drugs, left toe(s)                                                                    |
|          9 | Title 9      | Exposure to flames in controlled fire, not in building or structure                                        |
|         10 | Title 10     | Malignant otitis externa, left ear                                                                         |
|         11 | Title 11     | Other fracture of lower end of unspecified ulna, subsequent encounter for open fracture type I or II with delayed healing |
|         12 | Title 12     | Contusion and laceration of left cerebrum with loss of consciousness of 31 minutes to 59 minutes, initial encounter |
|         13 | Title 13     | Toxic effect of other corrosive organic compounds, accidental (unintentional), sequela                    |
|         14 | Title 14     | Laceration without foreign body of unspecified part of head                                                |
|         15 | Title 15     | Other chronic osteomyelitis, tibia and fibula                                                              |
+------------+--------------+------------------------------------------------------------------------------------------------------------+
15 rows in set (0.00 sec)

mysql>
```

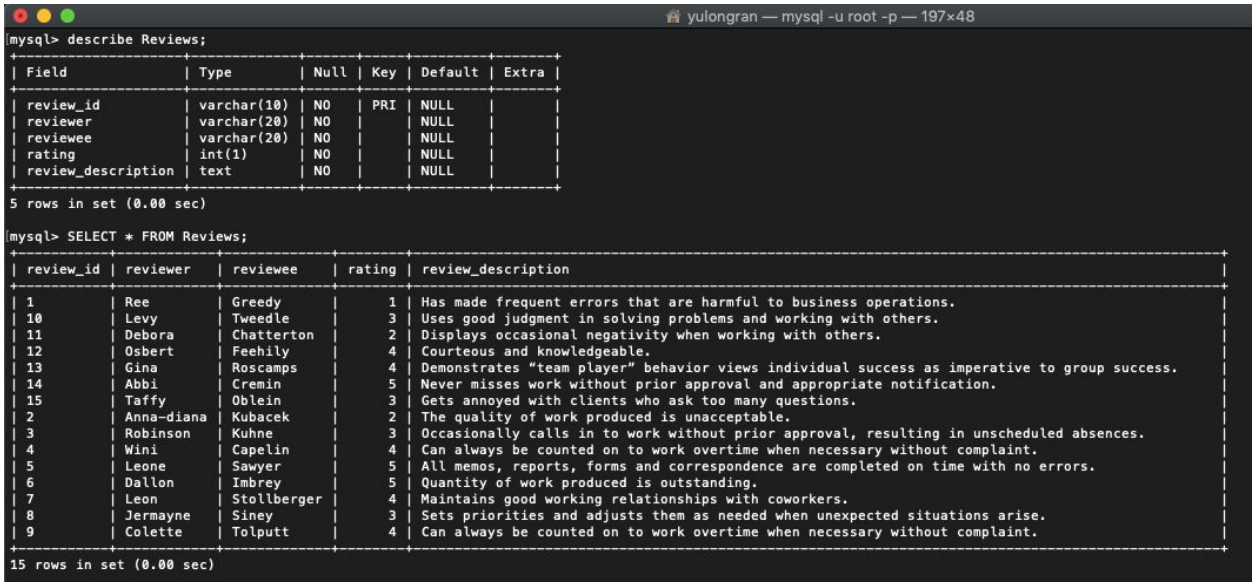
Milestones Table



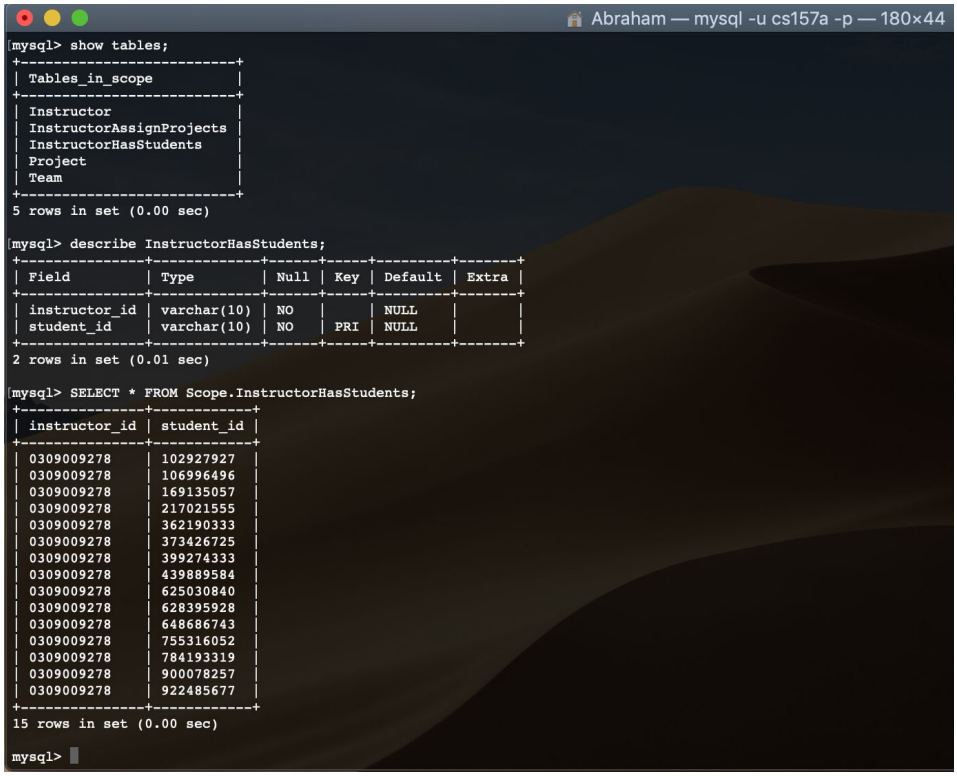
Team Table



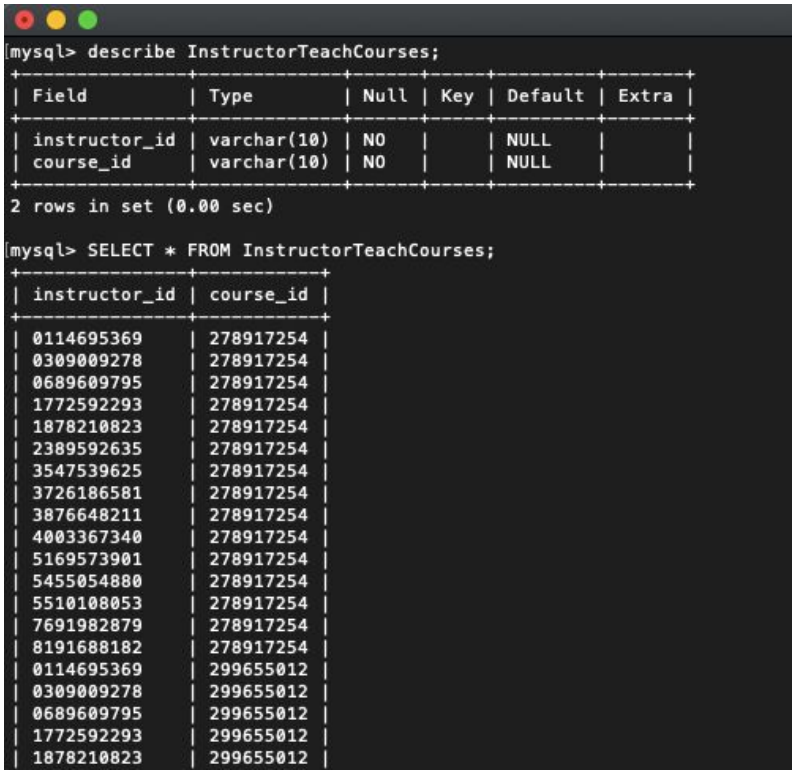
Reviews Table



InstructorHasStudent Table

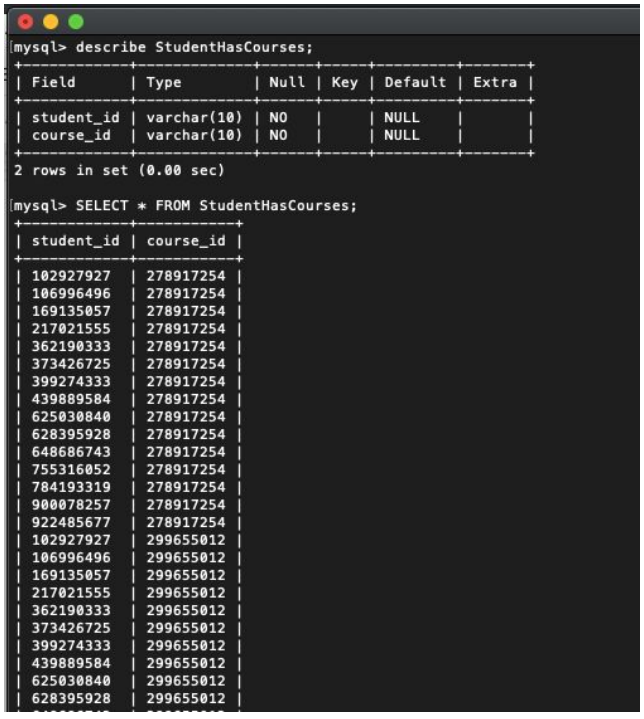


InstructorTeachCourses Table



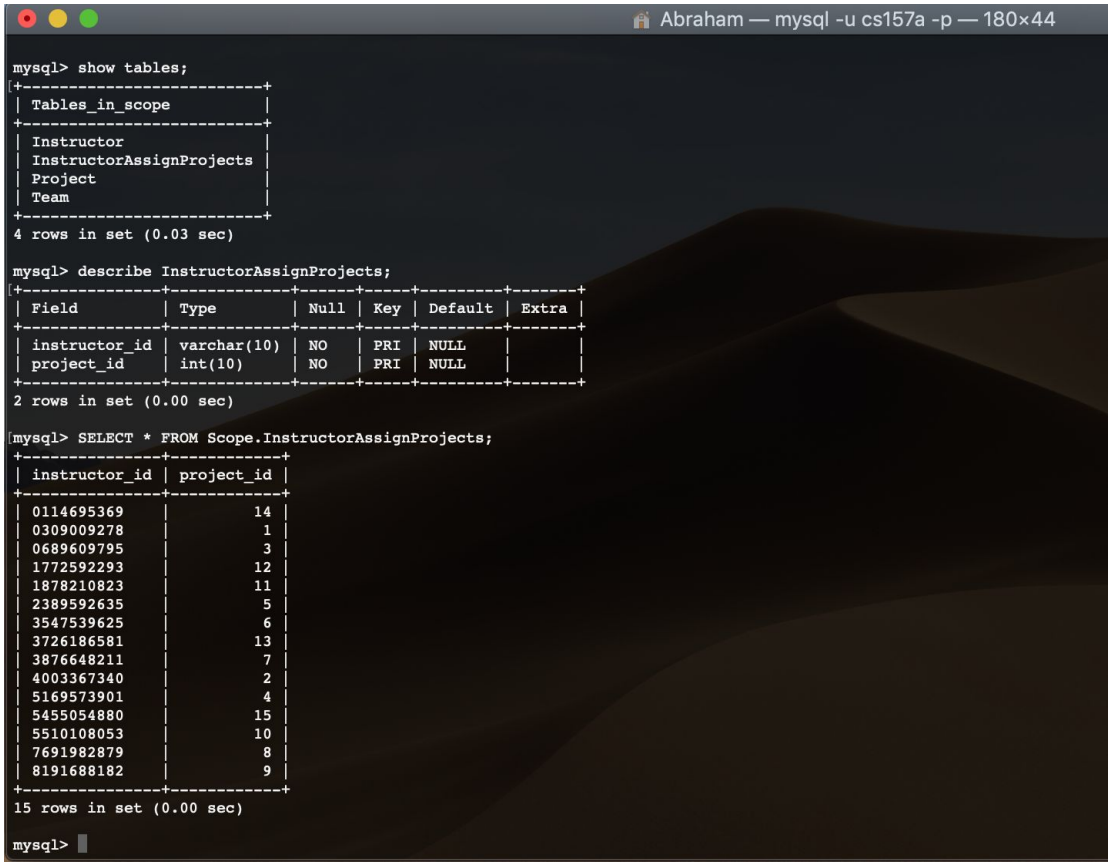
```
mysql> describe InstructorTeachCourses;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| instructor_id | varchar(10) | NO   |     | NULL    |       |
| course_id     | varchar(10) | NO   |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM InstructorTeachCourses;
+---------------+------------+
| instructor_id | course_id |
+---------------+------------+
| 0114695369    | 278917254 |
| 0309009278    | 278917254 |
| 0689609795    | 278917254 |
| 1772592293    | 278917254 |
| 1878210823    | 278917254 |
| 2389592635    | 278917254 |
| 3547539625    | 278917254 |
| 3726186581    | 278917254 |
| 3876648211    | 278917254 |
| 4003367340    | 278917254 |
| 5169573901    | 278917254 |
| 5455054880    | 278917254 |
| 5510108053    | 278917254 |
| 7691982879    | 278917254 |
| 8191688182    | 278917254 |
| 0114695369    | 299655012 |
| 0309009278    | 299655012 |
| 0689609795    | 299655012 |
| 1772592293    | 299655012 |
| 1878210823    | 299655012 |
```

StudentHasCourses Table

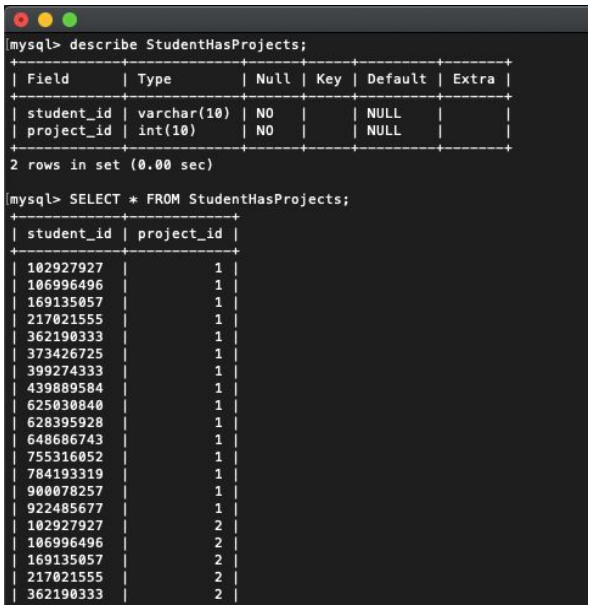


```
mysql> describe StudentHasCourses;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| student_id | varchar(10) | NO   |     | NULL    |       |
| course_id  | varchar(10) | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM StudentHasCourses;
+------------+-----------+
| student_id | course_id |
+------------+-----------+
| 102927927  | 278917254 |
| 106996496  | 278917254 |
| 169135057  | 278917254 |
| 217021555  | 278917254 |
| 362190333  | 278917254 |
| 373426725  | 278917254 |
| 399274333  | 278917254 |
| 439889584  | 278917254 |
| 625030840  | 278917254 |
| 628395928  | 278917254 |
| 648686743  | 278917254 |
| 755316052  | 278917254 |
| 784193319  | 278917254 |
| 900078257  | 278917254 |
| 922485677  | 278917254 |
| 102927927  | 299655012 |
| 106996496  | 299655012 |
| 169135057  | 299655012 |
| 217021555  | 299655012 |
| 362190333  | 299655012 |
| 373426725  | 299655012 |
| 399274333  | 299655012 |
| 439889584  | 299655012 |
| 625030840  | 299655012 |
| 628395928  | 299655012 |
```

InstructorAssignProjects Table



StudentHasProjects Table



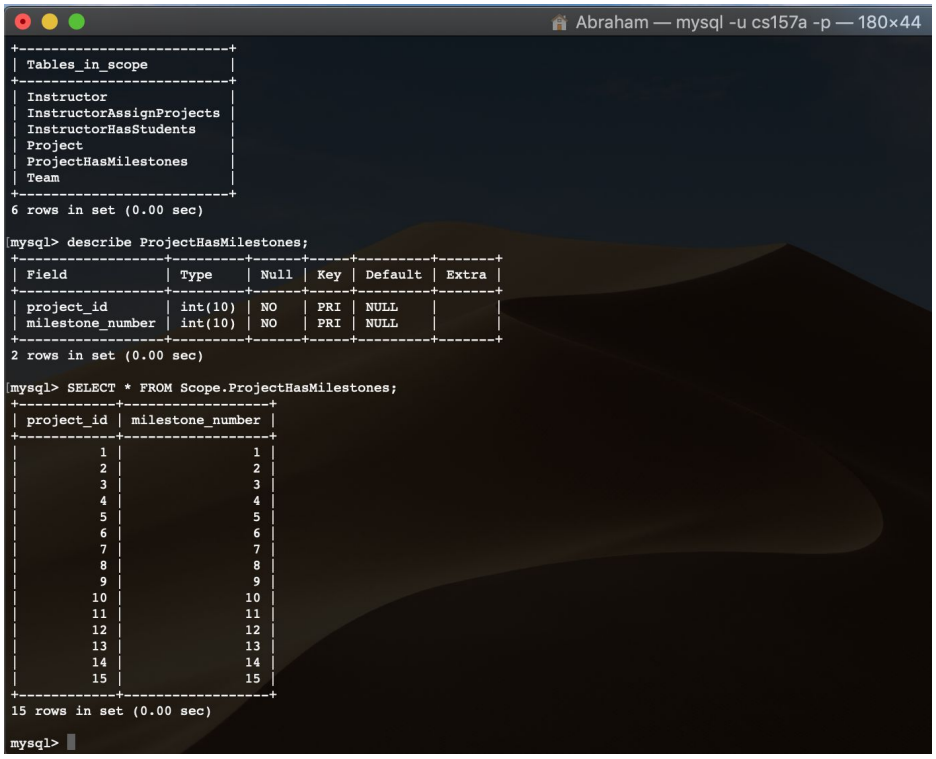
StudentHasTeams Table

```
mysql> describe StudentHasTeams;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| team_number  | int(3)      | NO   |     | NULL    |       |
| student_id   | varchar(10) | NO   |     | NULL    |       |
| project_id   | int(10)     | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM StudentHasTeams;
+-------------+------------+------------+
| team_number | student_id | project_id |
+-------------+------------+------------+
|           1 | 102927927  |          1 |
|           2 | 102927927  |          1 |
|           3 | 102927927  |          1 |
|           4 | 102927927  |          1 |
|           5 | 102927927  |          1 |
|           6 | 102927927  |          1 |
|           7 | 102927927  |          1 |
|           8 | 102927927  |          1 |
|           9 | 102927927  |          1 |
|          10 | 102927927  |          1 |
|          11 | 102927927  |          1 |
|          12 | 102927927  |          1 |
|          13 | 102927927  |          1 |
|          14 | 102927927  |          1 |
|          15 | 102927927  |          1 |
|           1 | 106996496  |          1 |
|           2 | 106996496  |          1 |
|           3 | 106996496  |          1 |
|           4 | 106996496  |          1 |
```

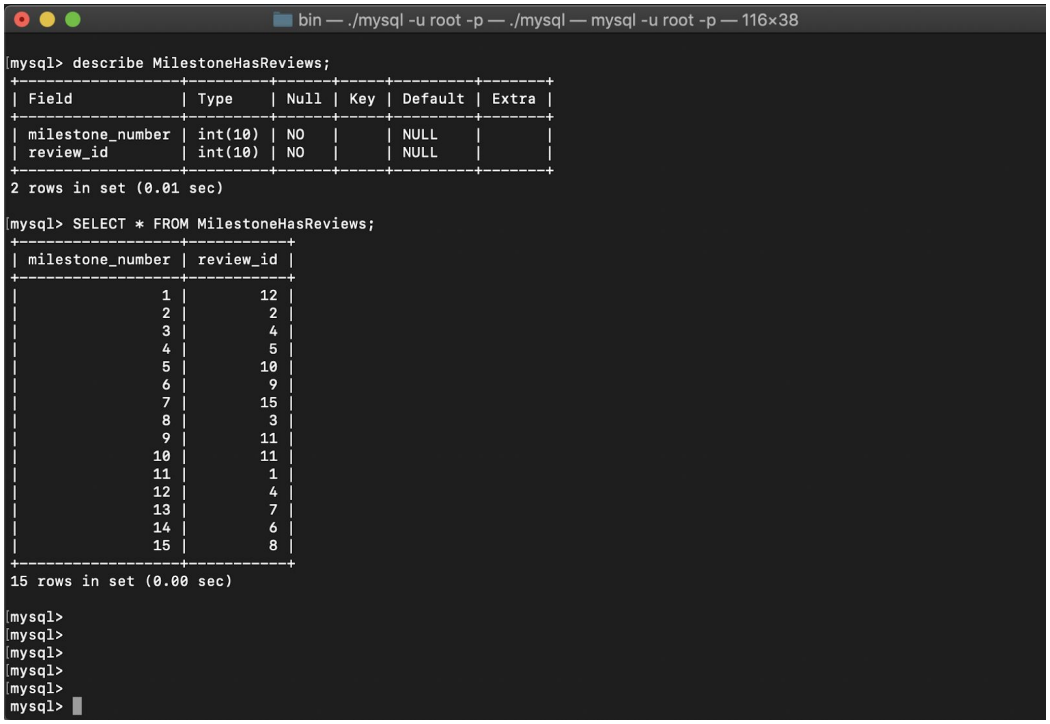
CoursesHasProjects Table

```
bin — ./mysql -u root -p — ./mysql — mysql -u root -p — 116×38

mysql> describe CoursesHasProjects;
+------------+---------+------+-----+---------+-------+
| Field      | Type    | Null | Key | Default | Extra |
+------------+---------+------+-----+---------+-------+
| course_id  | int(10) | NO   |     | NULL    |       |
| project_id | int(10) | NO   |     | NULL    |       |
+------------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM Scope.CoursesHasProjects;
+-----------+------------+
| course_id | project_id |
+-----------+------------+
| 278917254 |          1 |
| 278917254 |          2 |
| 278917254 |          3 |
| 278917254 |          4 |
| 278917254 |          5 |
| 278917254 |          6 |
| 278917254 |          7 |
| 278917254 |          8 |
| 246924774 |          9 |
| 246924774 |         10 |
| 246924774 |         11 |
| 246924774 |         12 |
| 246924774 |         13 |
| 246924774 |         14 |
| 246924774 |         15 |
+-----------+------------+
15 rows in set (0.00 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
```

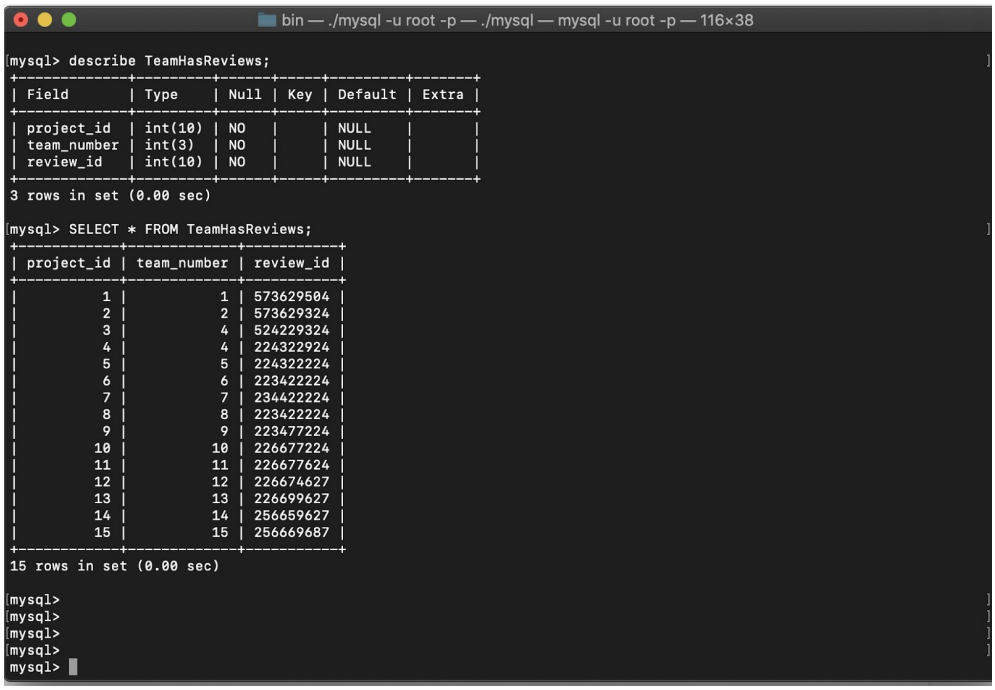
ProjectsHasMilestones Table



MilestoneHasReviews Table



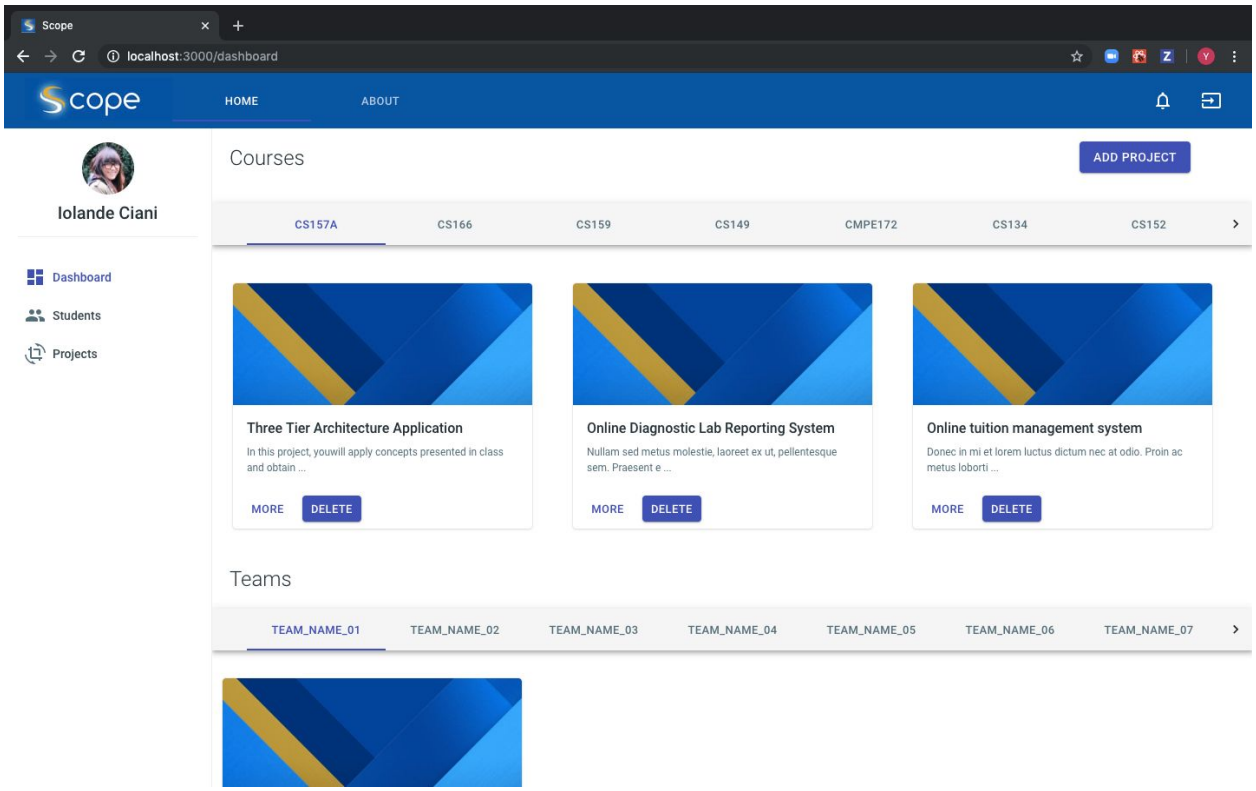
TeamHasReviews Table



Implementation

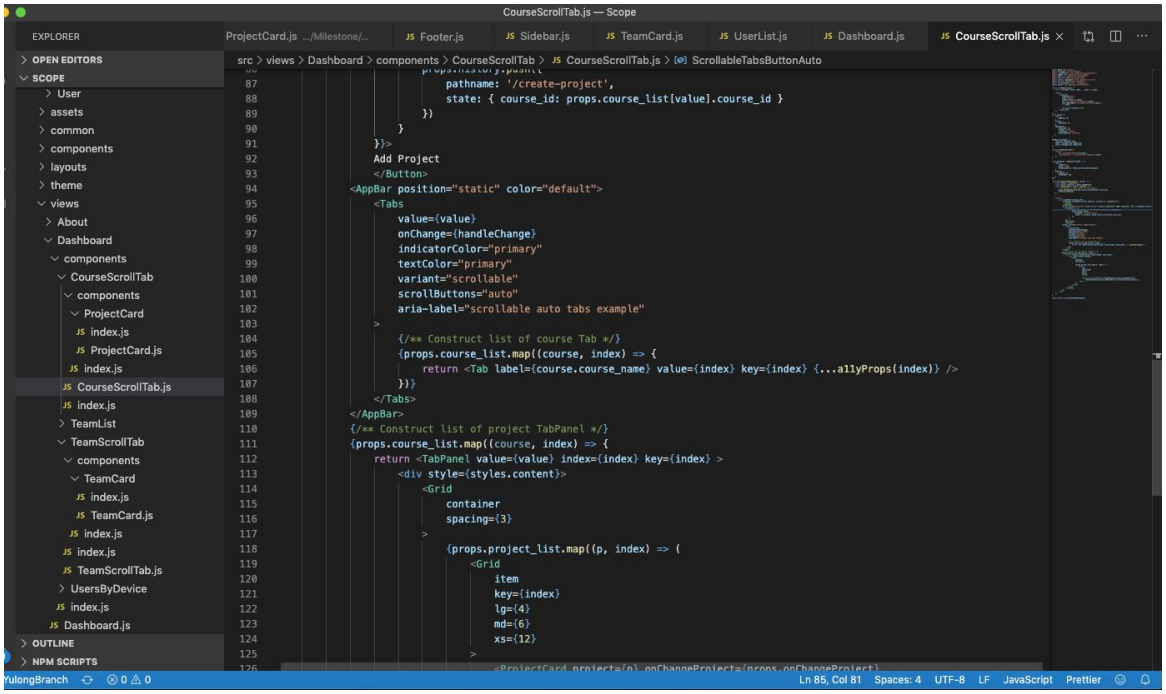
Display Course

Font-End



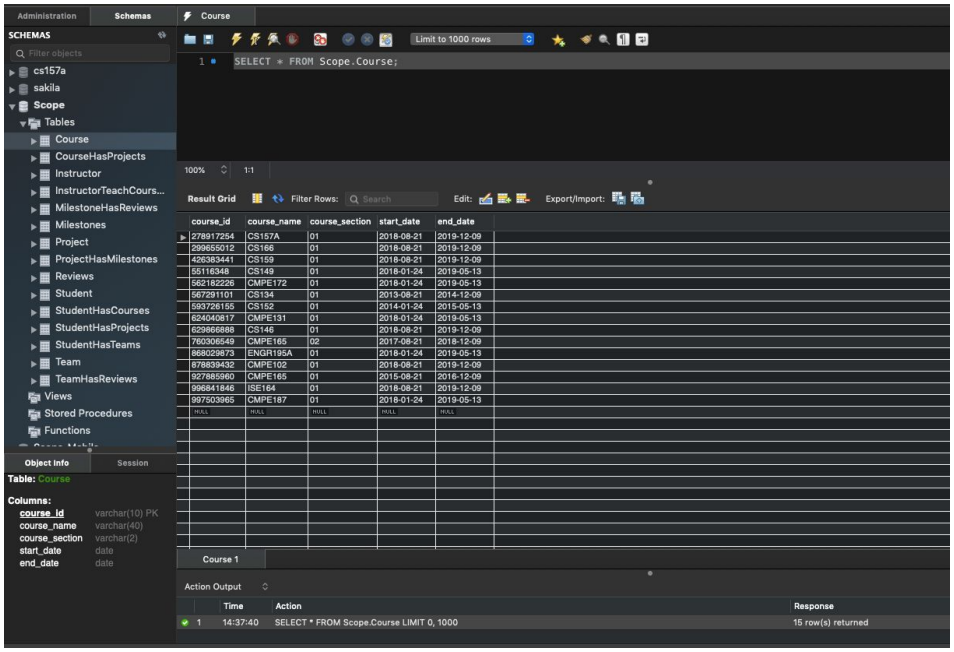
The above screenshot shows the dashboard of Scope. The dashboard page contains a CourseScollTab, which allows the user to select a different course.

Front-End Code

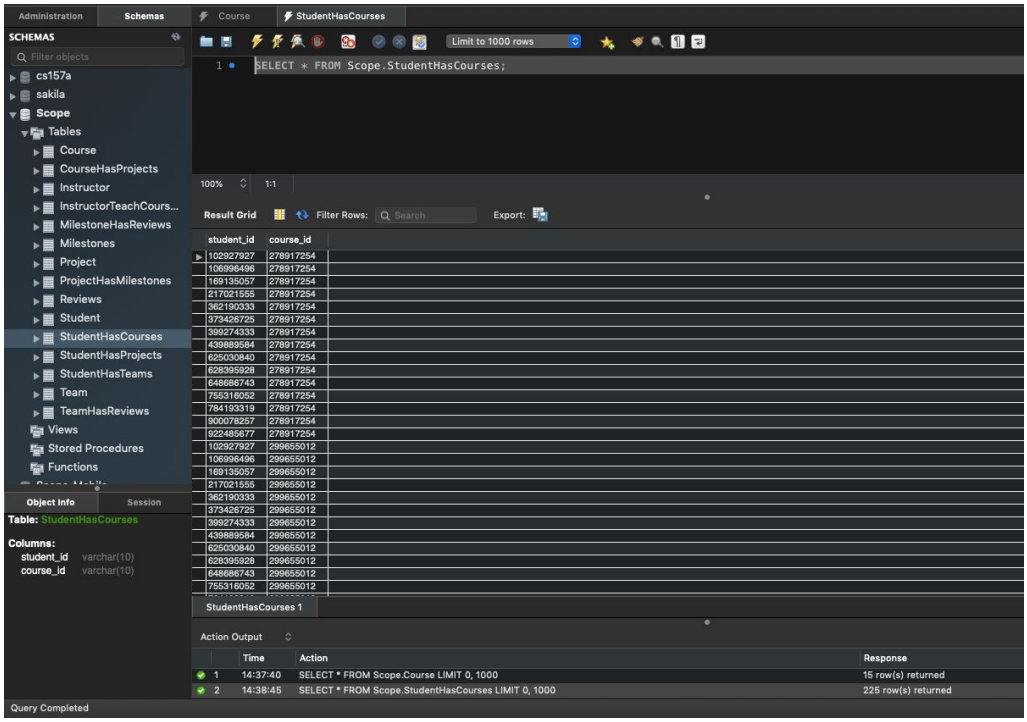


CourseScrollTab takes props from parent and creates tab for each course with title as course name.

Tables

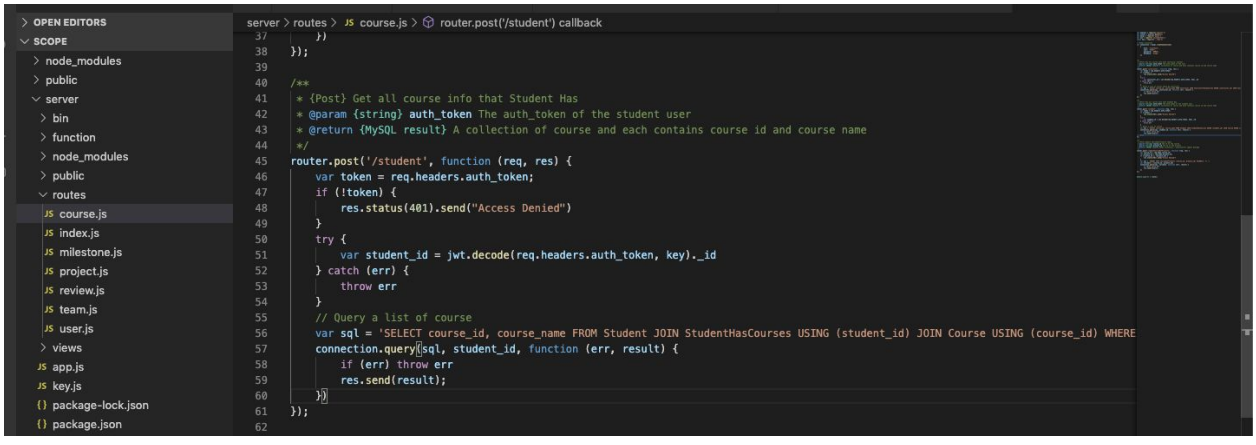


Course table in the database stored all course information.



User gets list of course through the table StundetHasCourse JOINS Course. User will send an API request to fetch all the course under a given student ID.

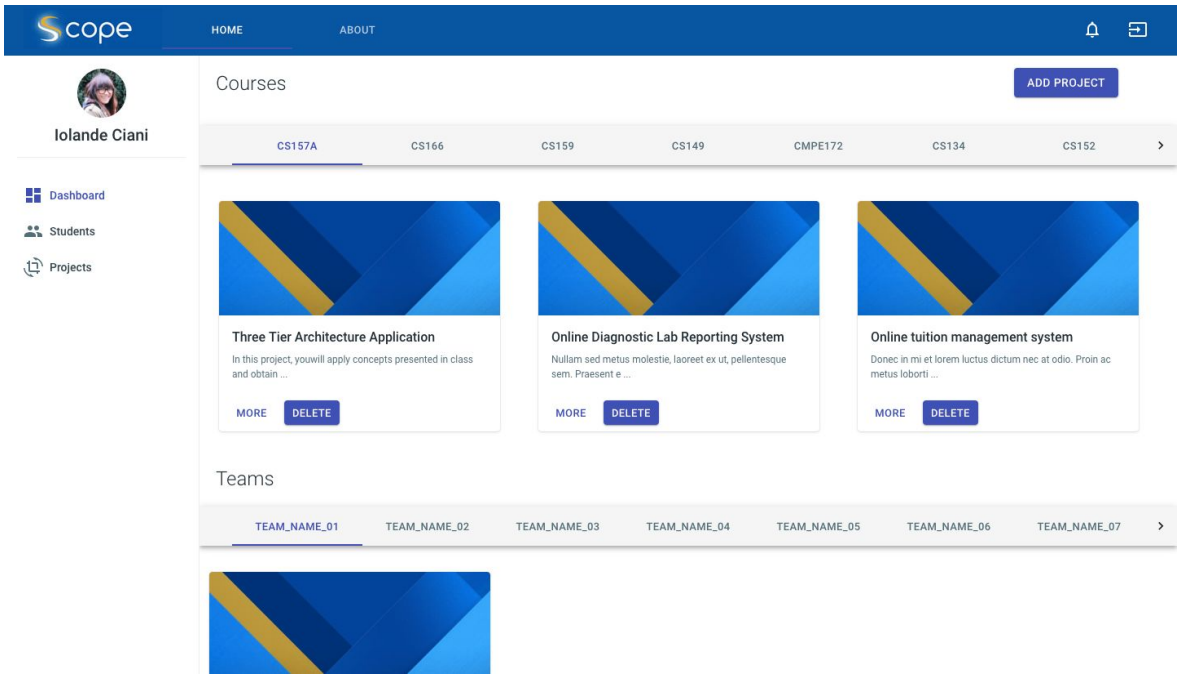
Backend Code



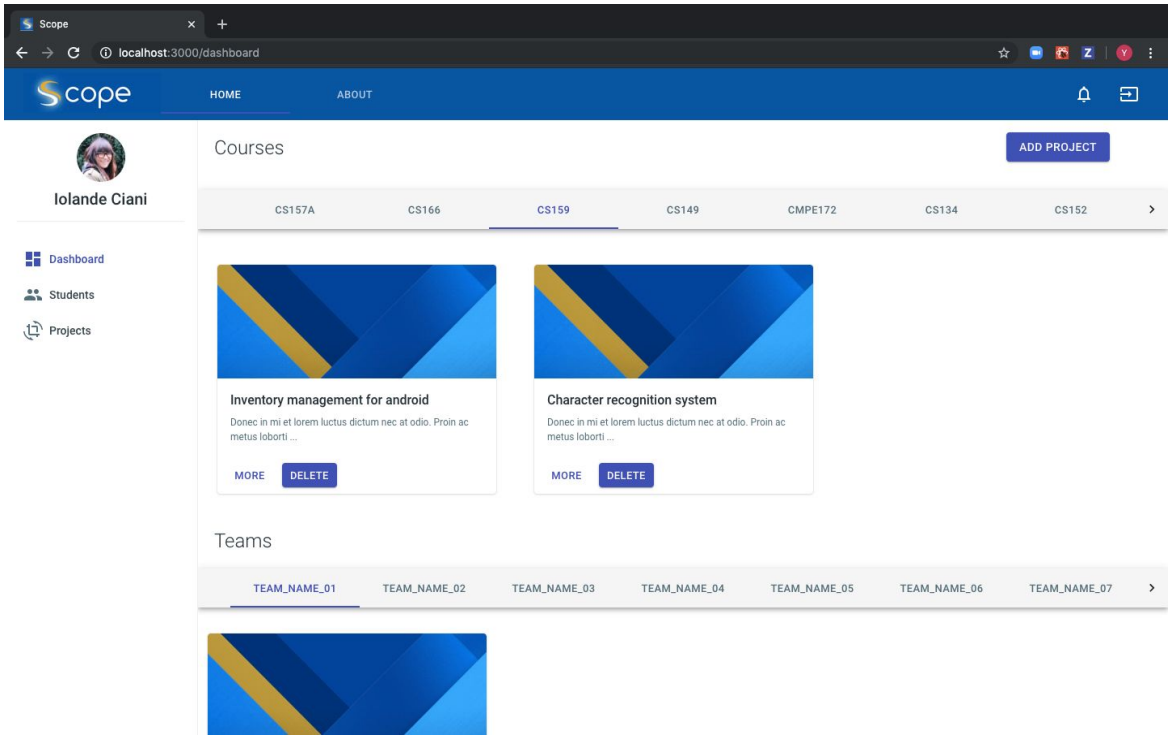
Above is the API call to fetch all the course based on the Student ID. The student ID will be sent by the User from the front-end as a JWT token.

Display Project Details

Front-End

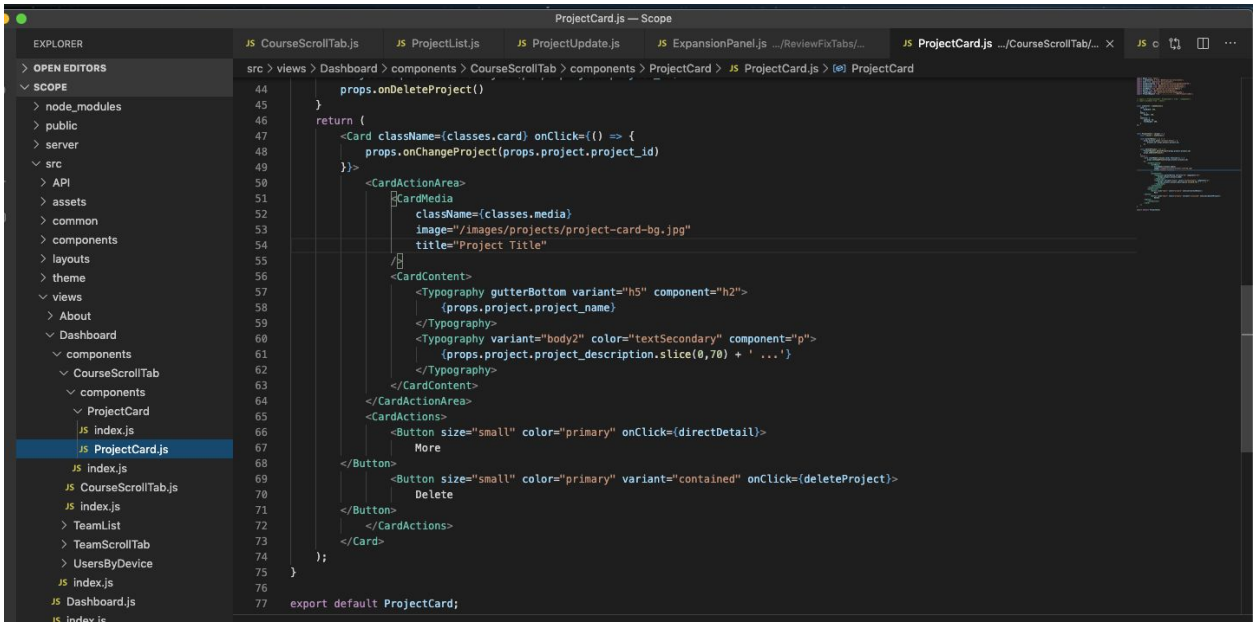


The above screenshot shows the project Information displays as a Project Card in the dashboard. Each Project card contains the name of the project, a description of the project, an option to delete the project, and view more details.



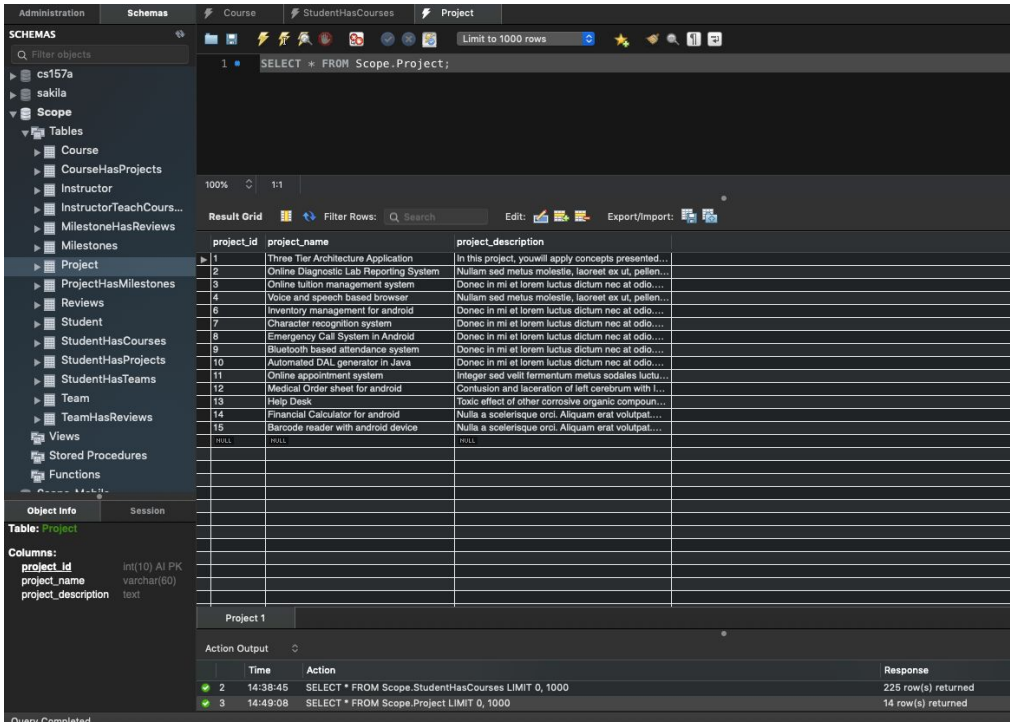
Clicking on different course tab in the CourseScrollTab will display a list of projects under that specific course.

Front-End Code

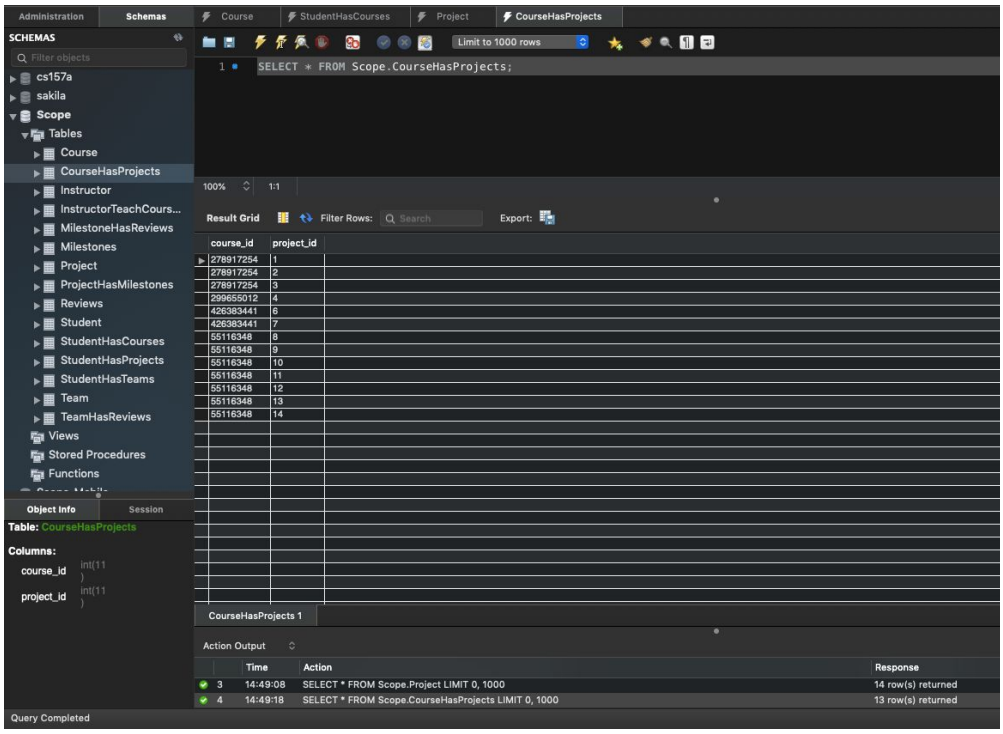


The above screenshot shows the corresponding front-end source code to receive project information from the parent CourseScrollTab as props.

Tables

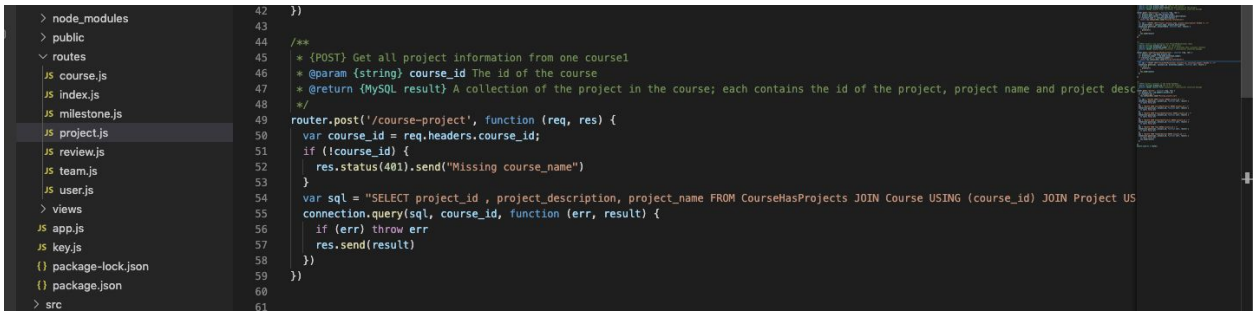


The Project Table stores all the project information in the Scope application.



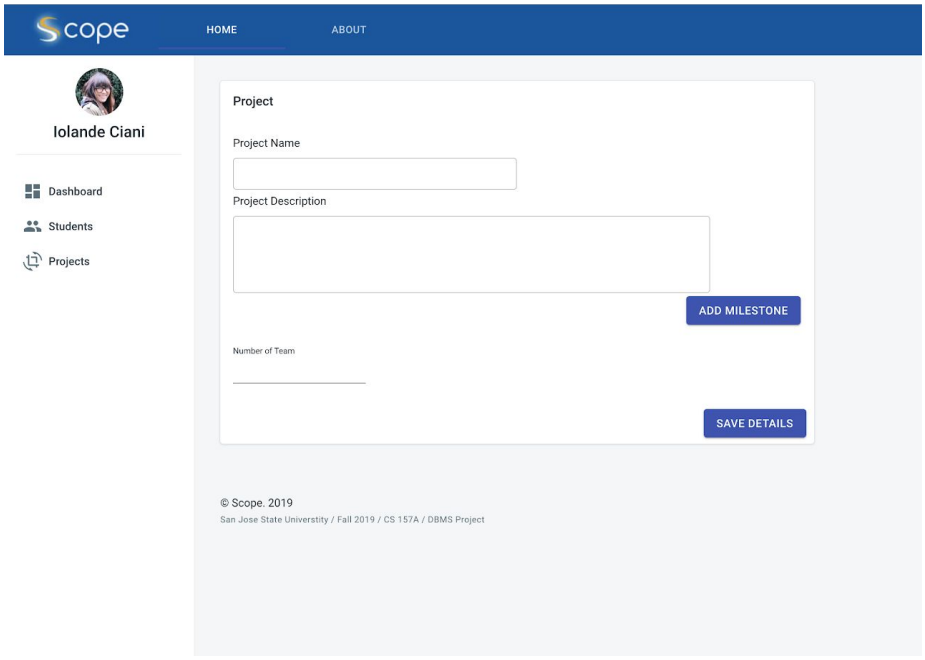
The Dashboard fetches all the project information based on the course_id from the Course collab. As shown above, the course_id 426383441 represents the Course CS159. It has two projects 6 and 7. From the project table, the projects with id 6 and 7 are Inventory management for android and Character recognition system.

Backend Code

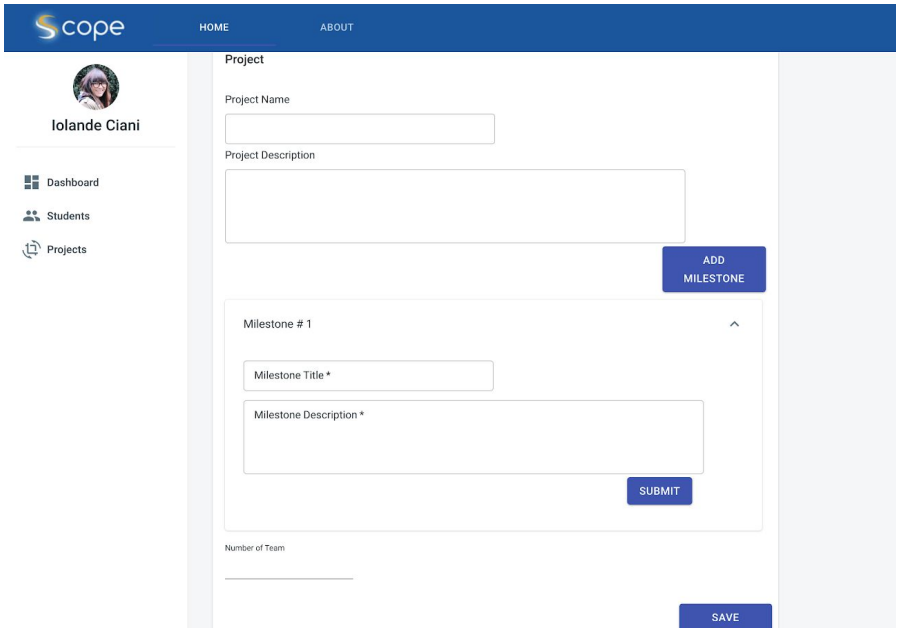


The above screenshot shows the corresponding API to fetch all projects information under specific course.

Add Project & Add Milestones
Front-End

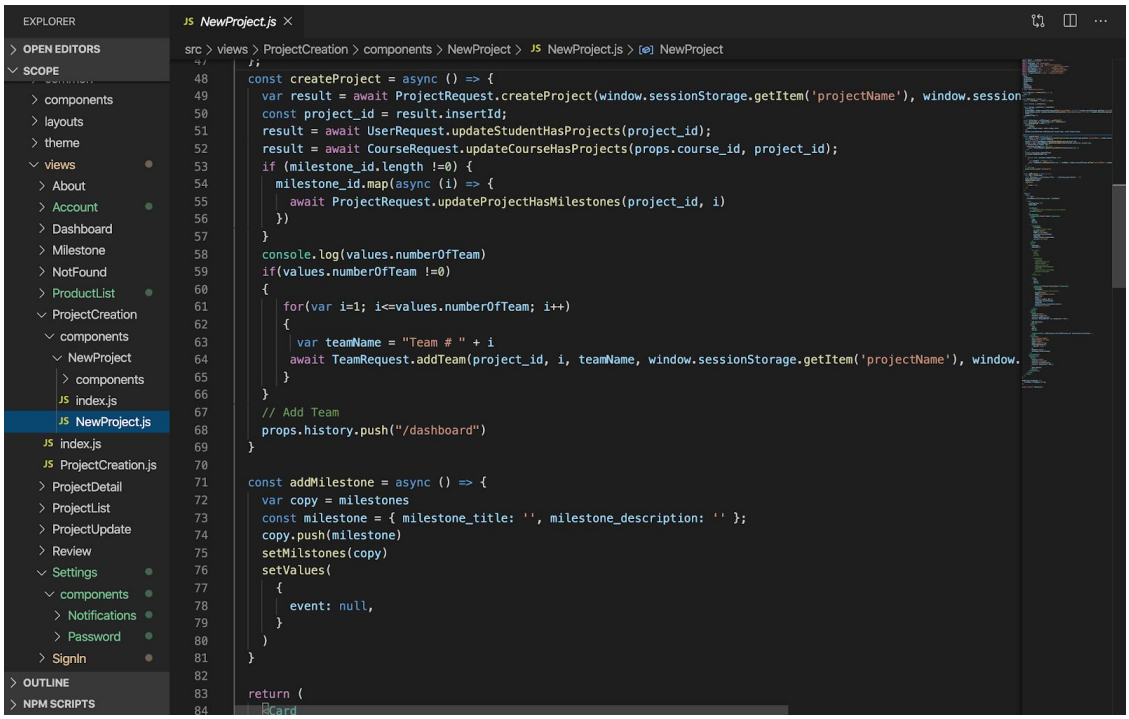


The above screenshot shows the pop up window that displays when the user clicks on the 'Add Project' button on the top right corner of the dashboard. This popup page contains a grid which includes text fields where the user can enter the project name, project description, add milestones, specify number of teams, and save the information.

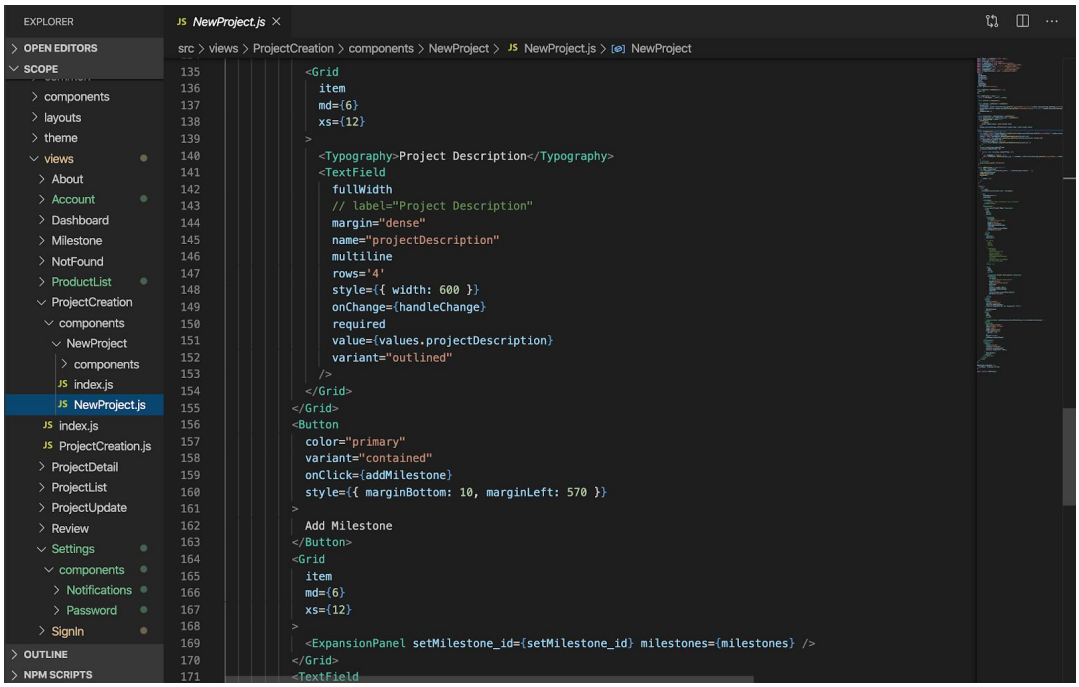


The above screenshot shows how the user can also add multiple milestones per project by clicking the ‘Add Milestone’ button add enter the title and description.

Front-End Code

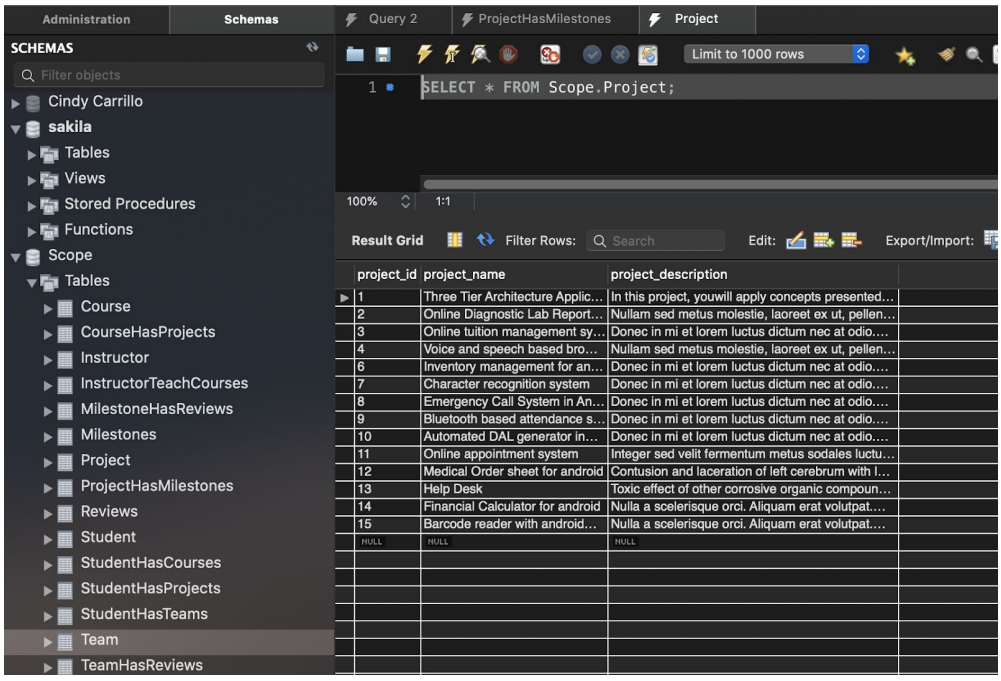


The above screenshot shows the corresponding front-end source code to create a project and milestones with specified information from the user.

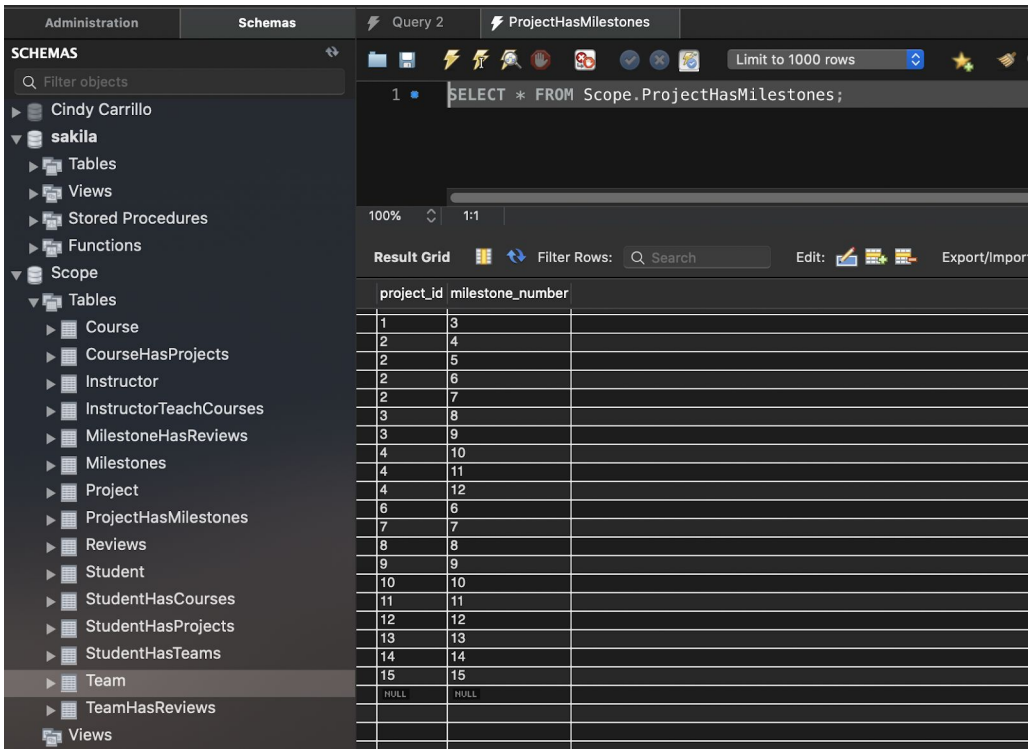


The above screenshot shows the corresponding front-end source code to organize the pop up window into sectioned grids for a clean UI.

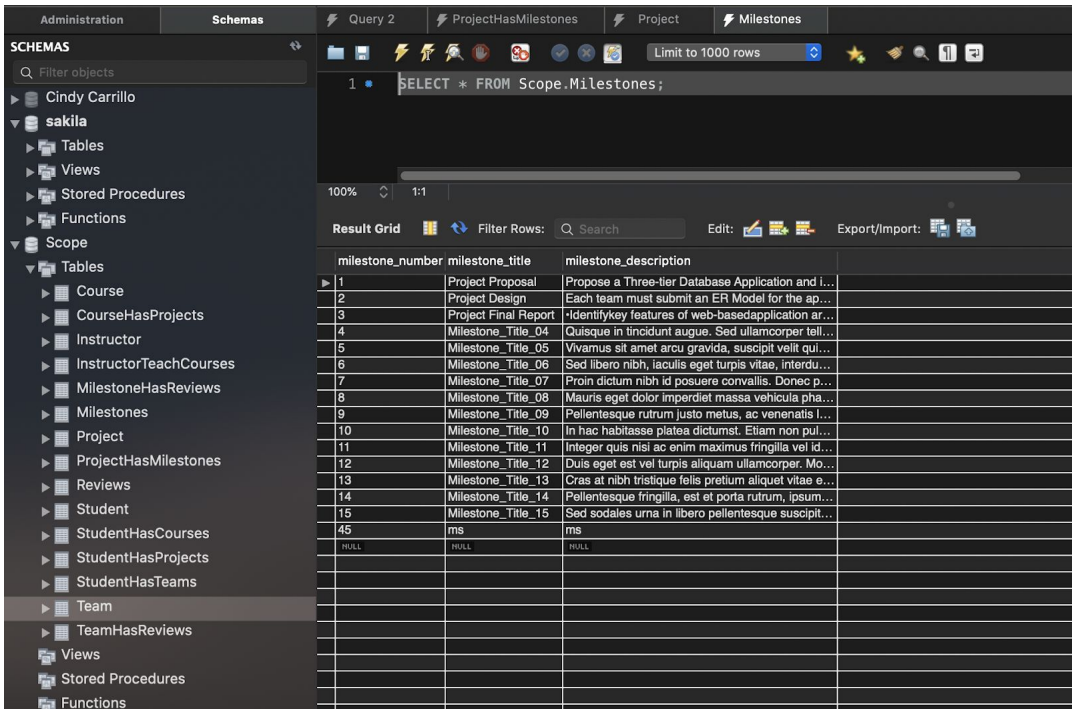
Tables



In the above screenshot, the Project Table stores all the project information: name and description, and specifies a unique project_id to each.

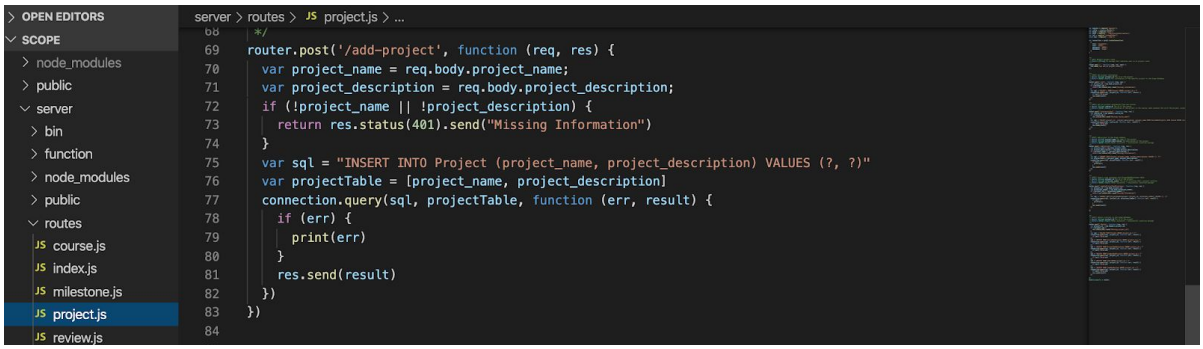


As shown above, the ProjectHasMilestones table shows the relationship between project and milestones. A project_id can be assigned to multiple milestone_numbers indicating that one project can have many milestones created. For example project_id = 1 has 3 milestones.

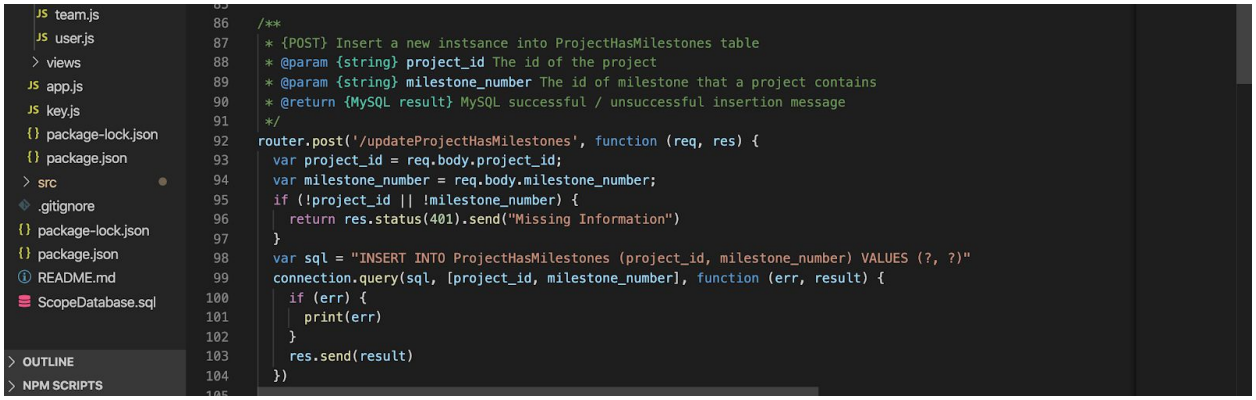


In the above screenshot, the Milestones table contains a title and a description tied to a unique milestone_number that will be used in the ProjectHasMilestone relationship table.

Backend Code

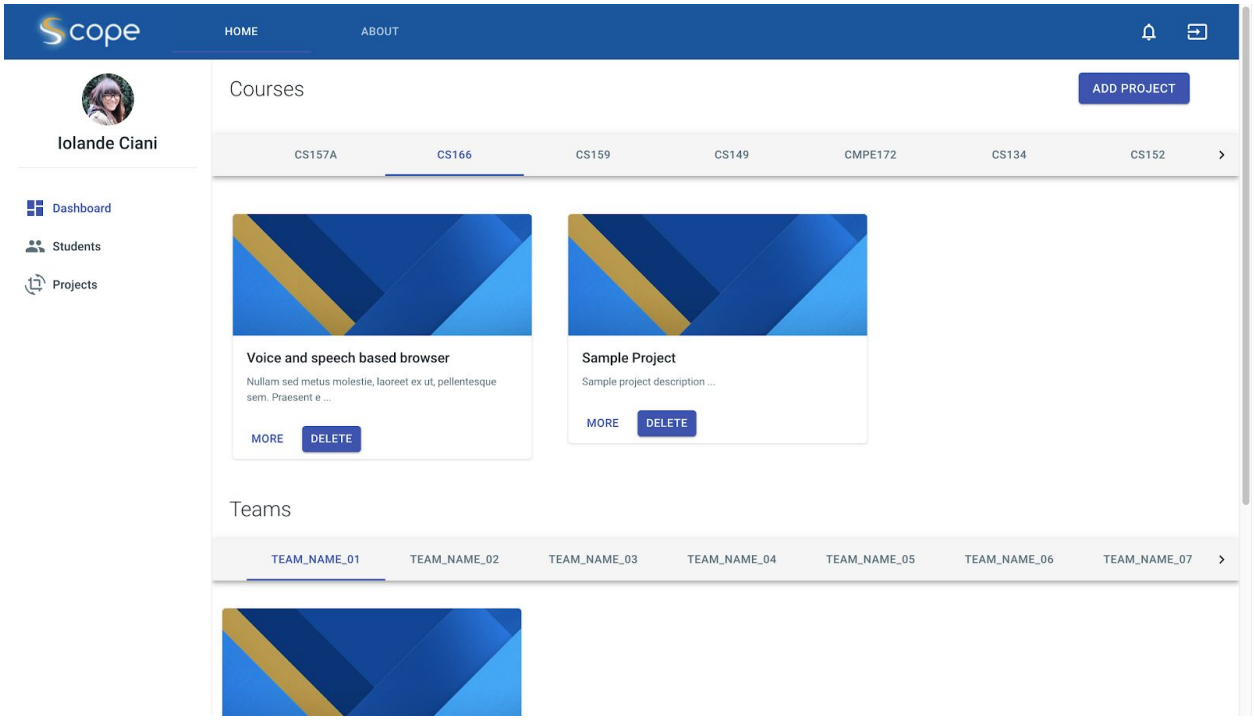


The above screenshot shows the corresponding API to add a project with specified information.

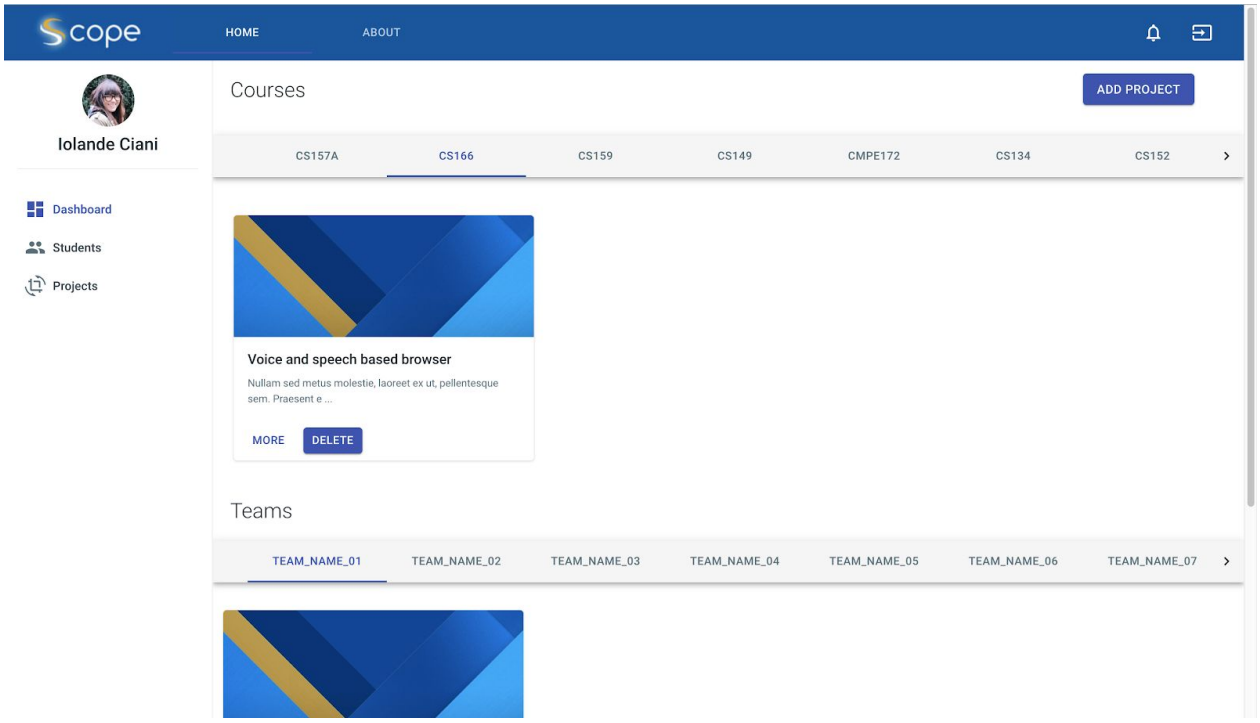


The above screenshot shows the corresponding API to update project information with a milestone that includes a title and description.

Delete Project
Front-End

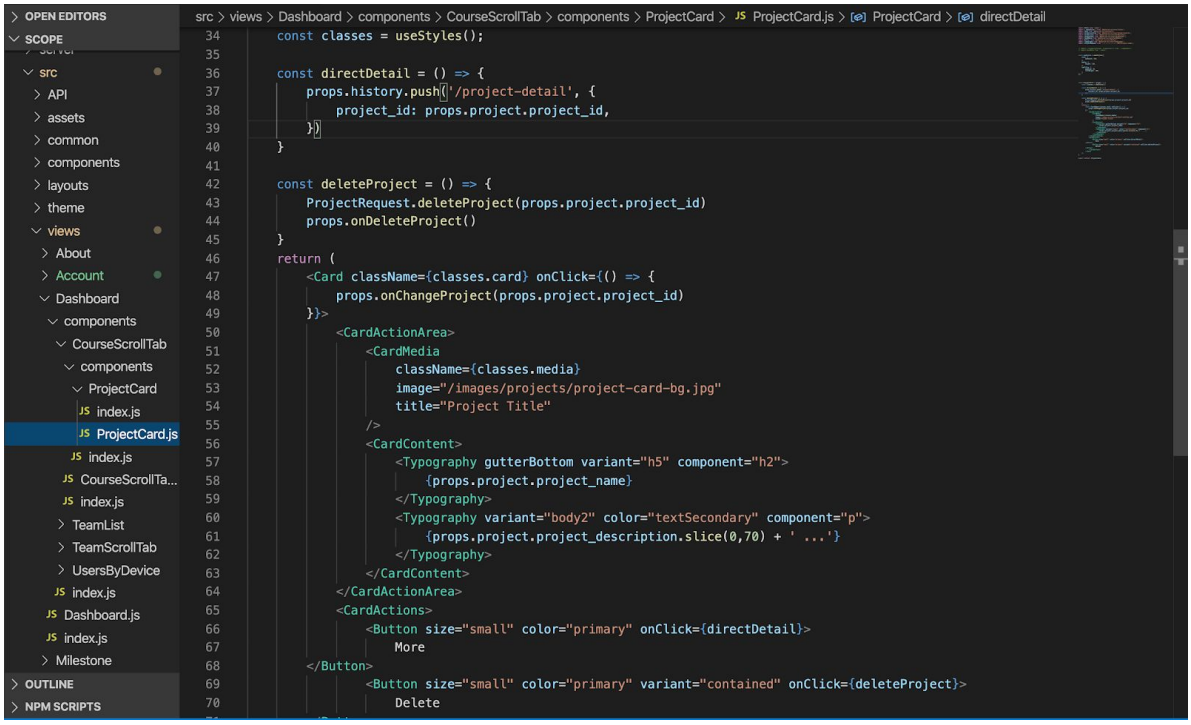


The above screenshot shows an added project in the CS166 course called ‘Sample Project’.



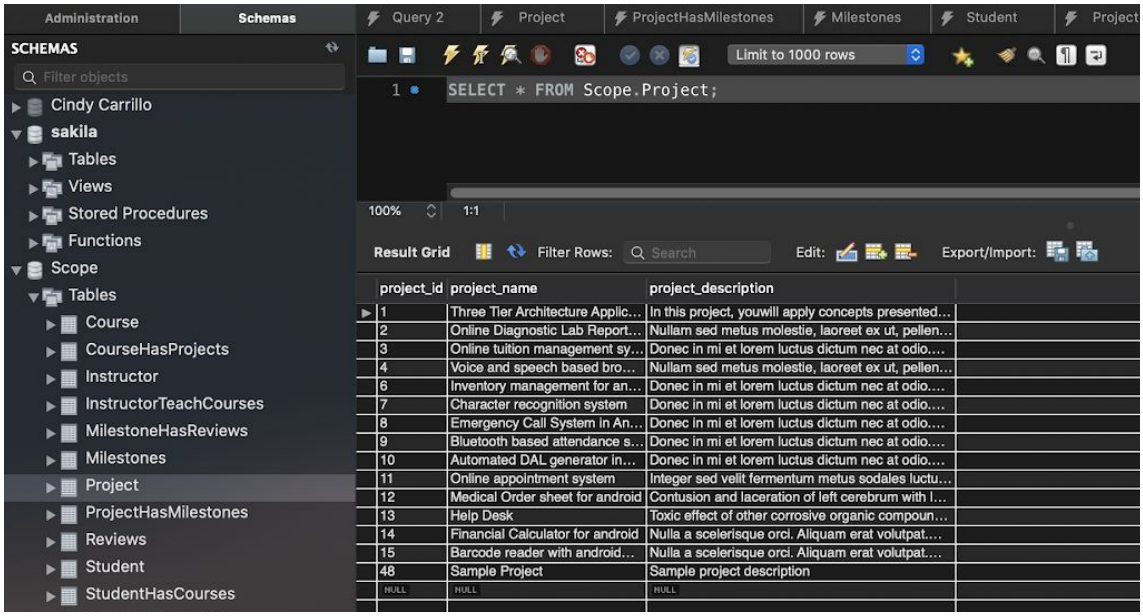
Once the delete button is clicked, the project is deleted from the course and removed from display.

Front-End Code

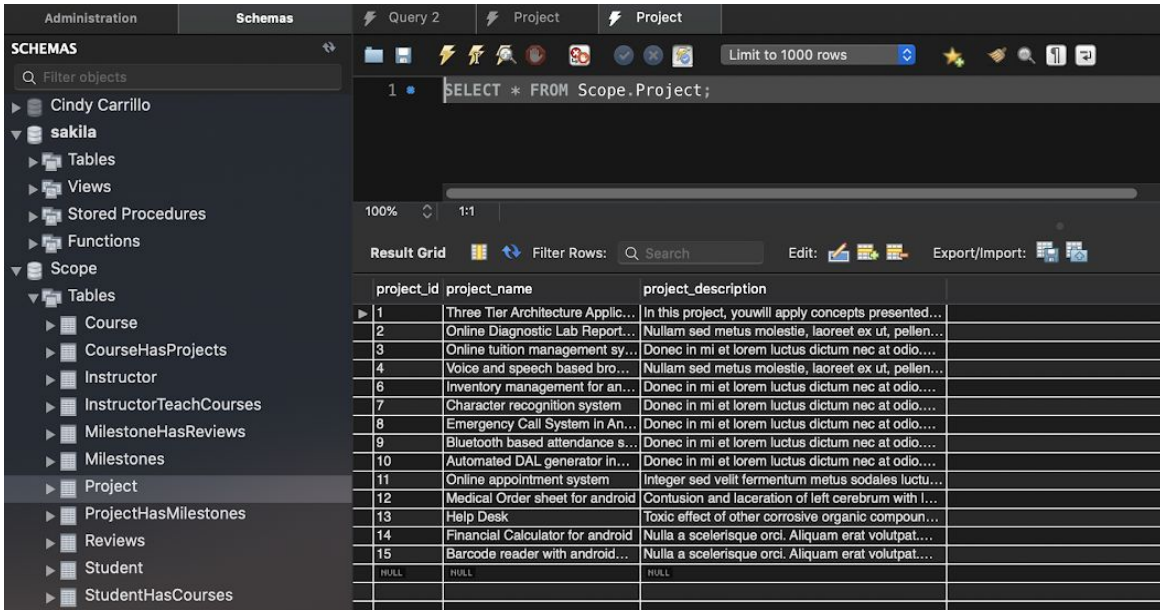


The above screenshot shows the deleteProject function call which deletes the project and the ‘Delete’ button specified in the CardActionArea.

Tables

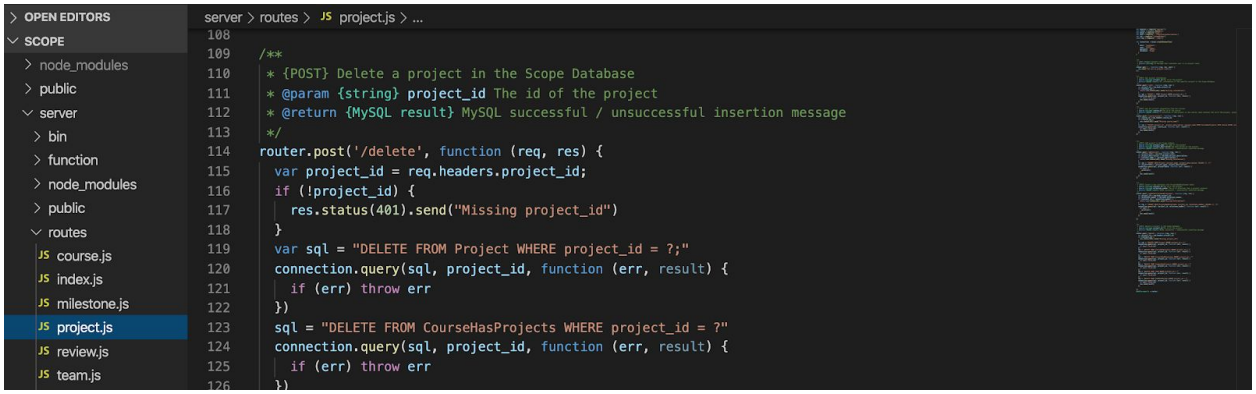


The above screenshot shows the Project table with the Sample Project added to it.



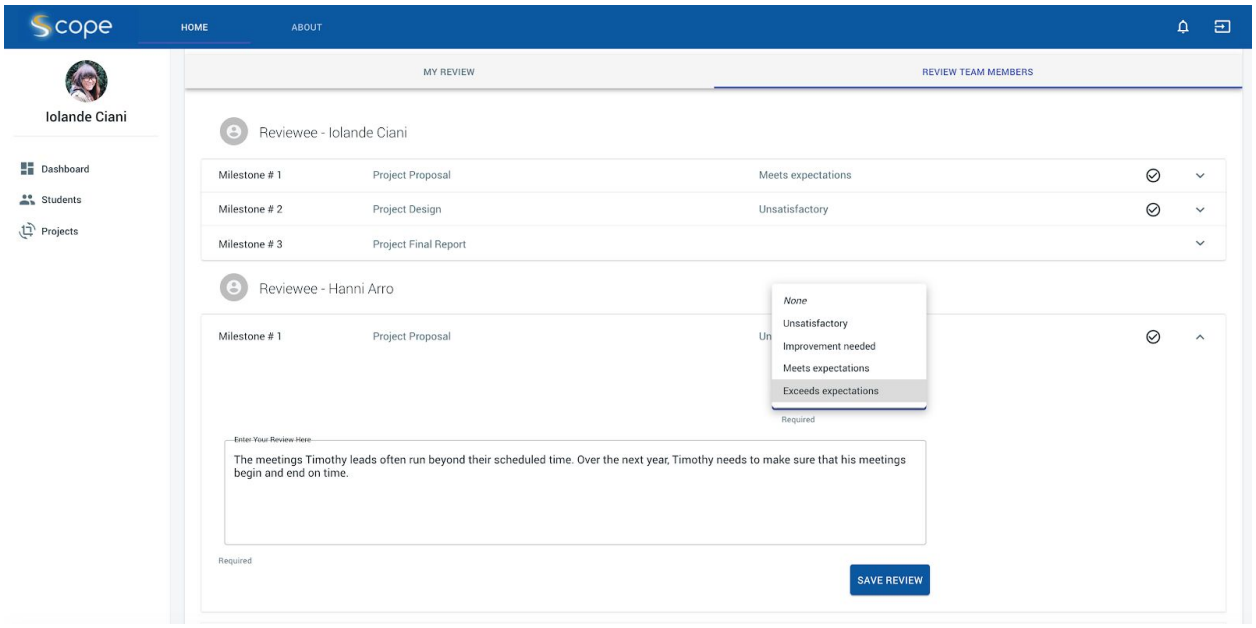
This screenshot shows that the Sample Project is deleted from the database once the delete button is clicked for that project.

Backend Code



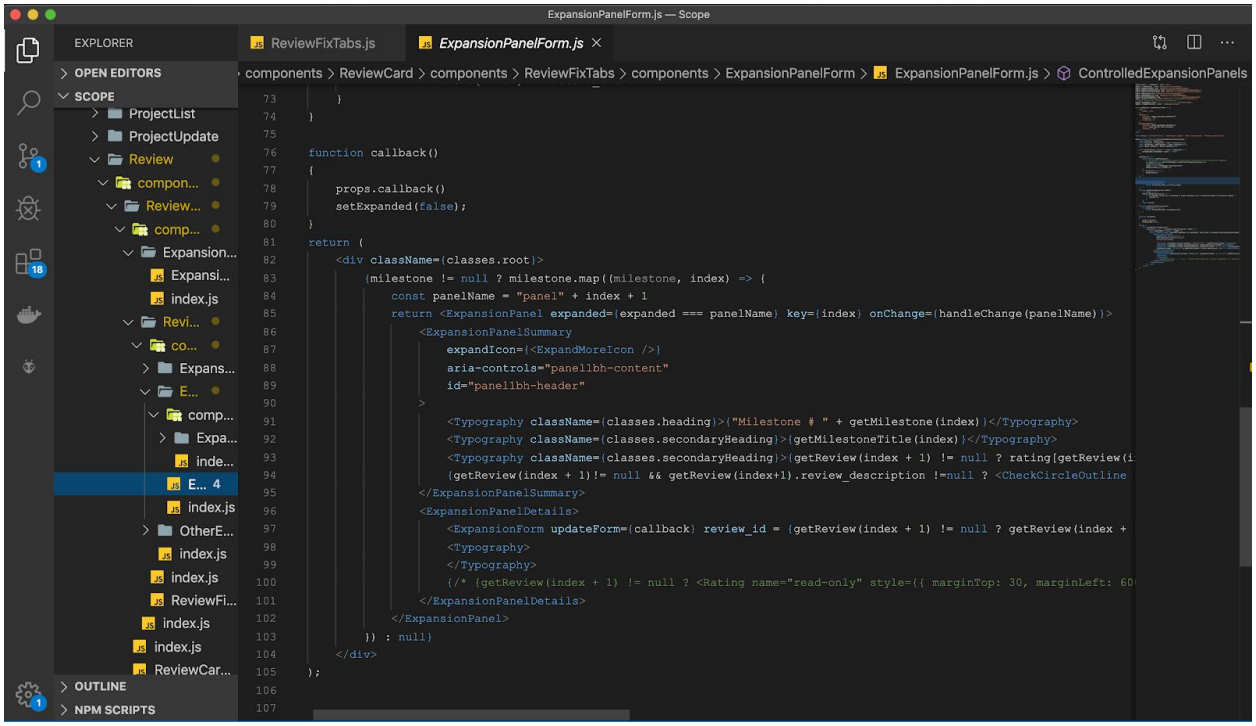
The above screenshot shows that the back end code for deleting a project and updating the database server.

Add Review
Front-End

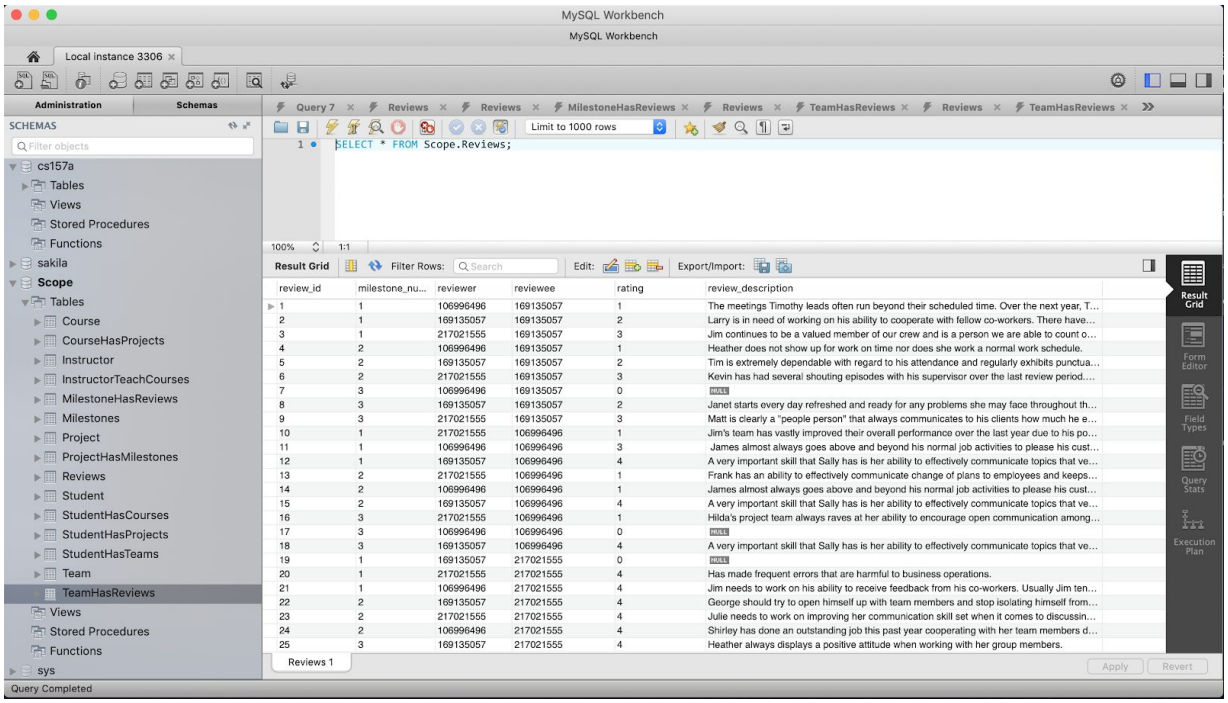


The above screenshot of Review Team Members shows all the team members the current student user can write project performance reviews. As you can see on the screenshot; once the user click Review Team Member navigation panel, it will pull all the reviewee student information along with each milestone to input the review and dropdown menu to select what performance review that student got.

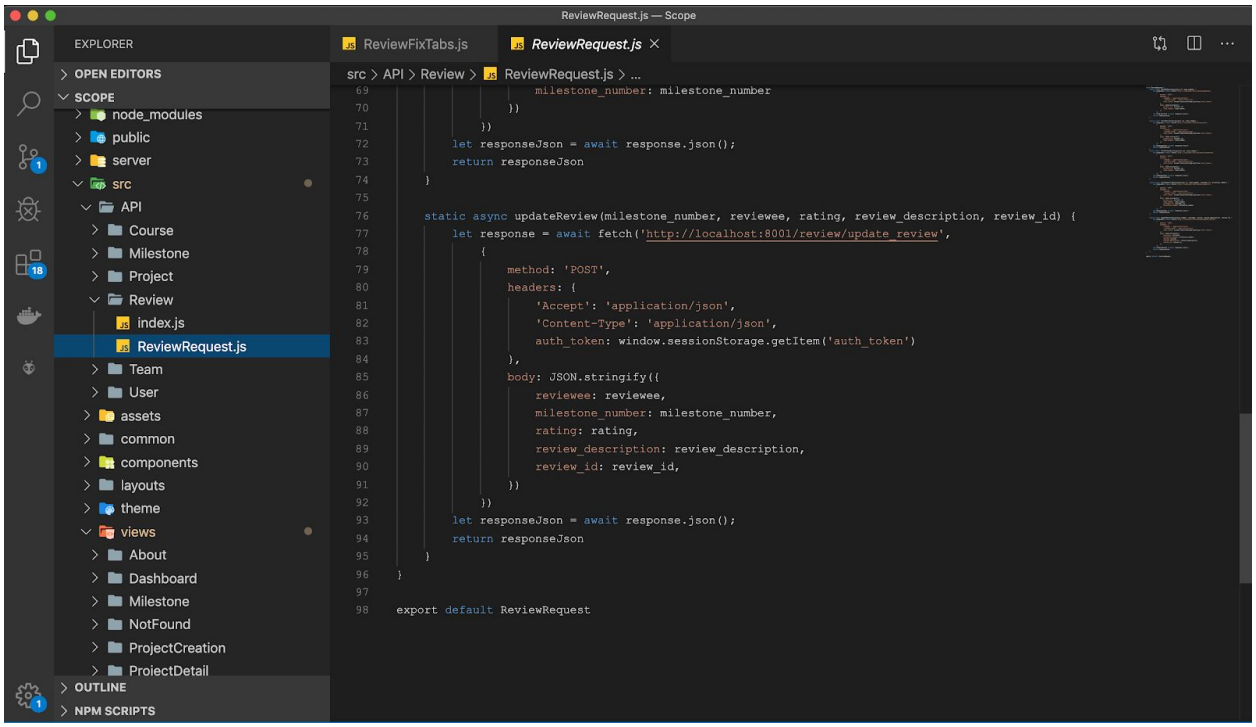
Front-End Code



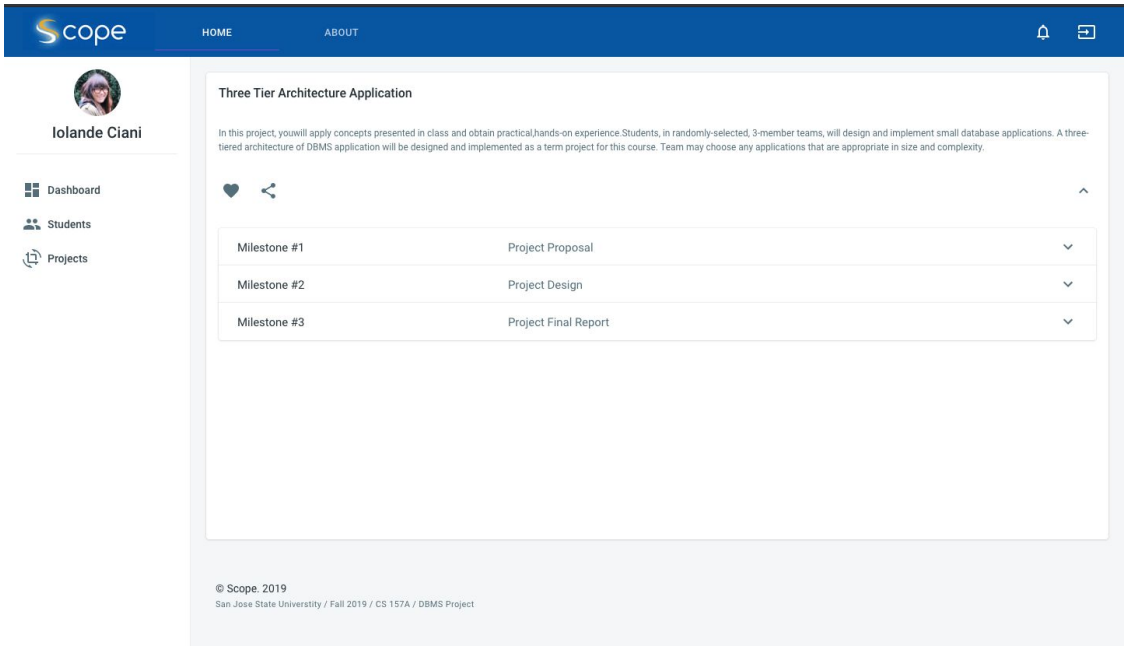
Tables



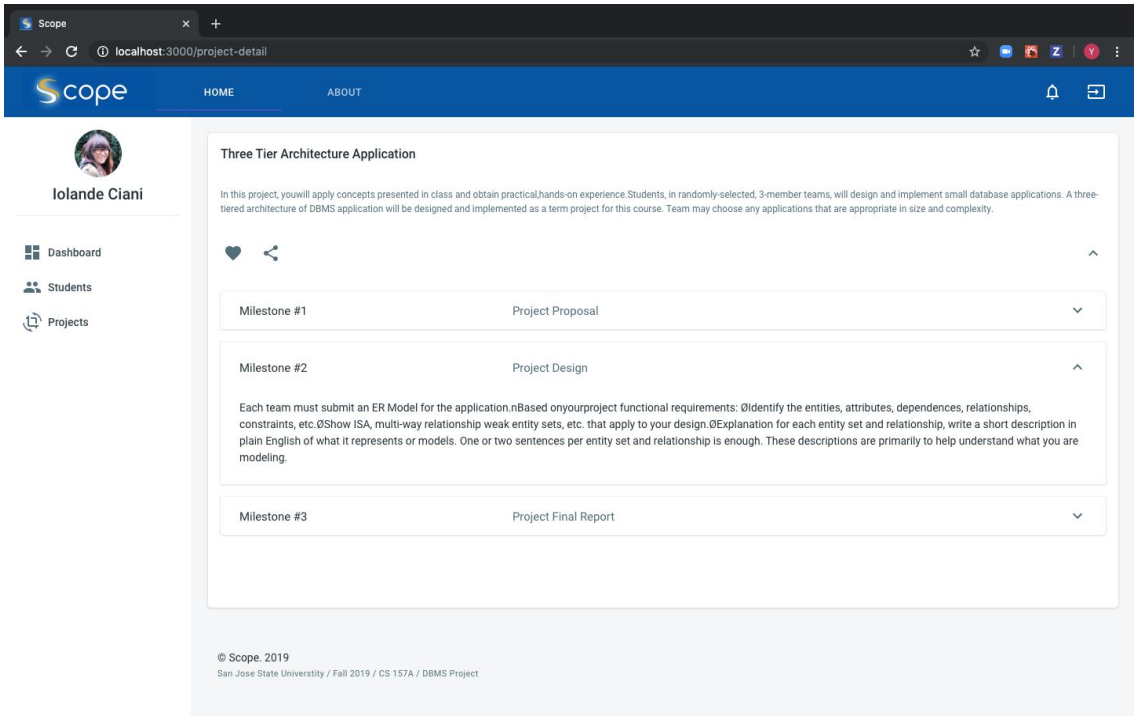
Backend Code



View Milestones
Front-End

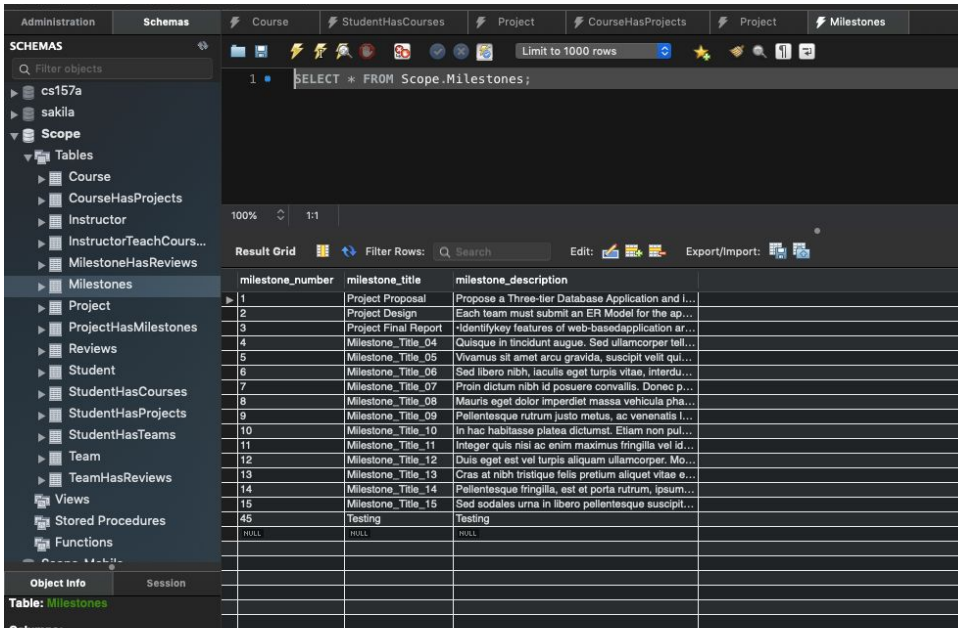


After clicking more button on the project card, Scope will direct to another page with Project Details. Project Detail page contains the name of the project, full description of the project and a list of milestones.

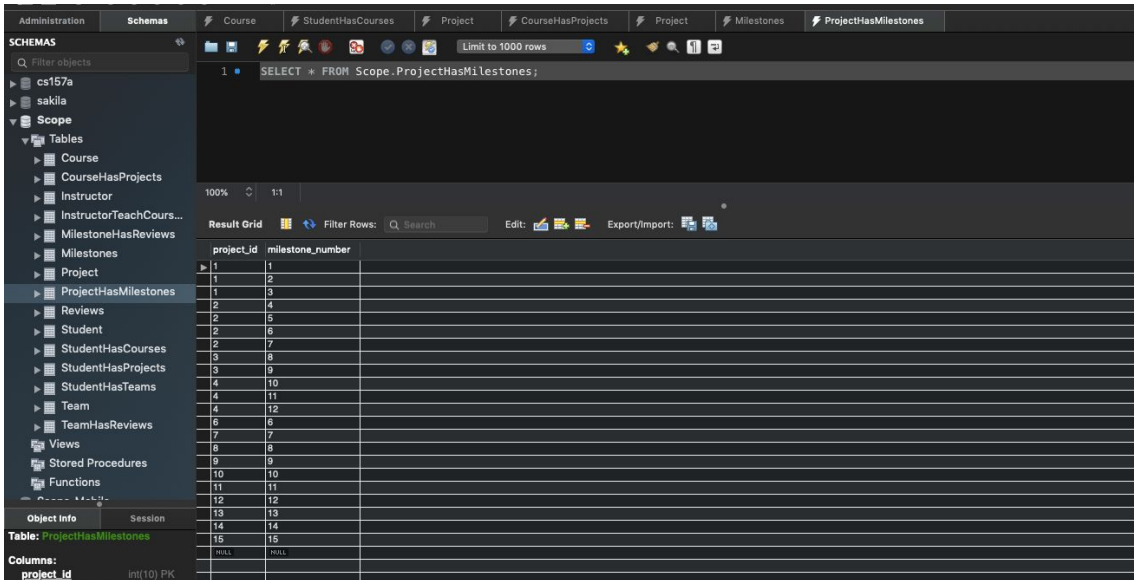


Each milestone dropdown menu contains a Milestone number, Milestone title and full description of the milestone from the database.

Tables



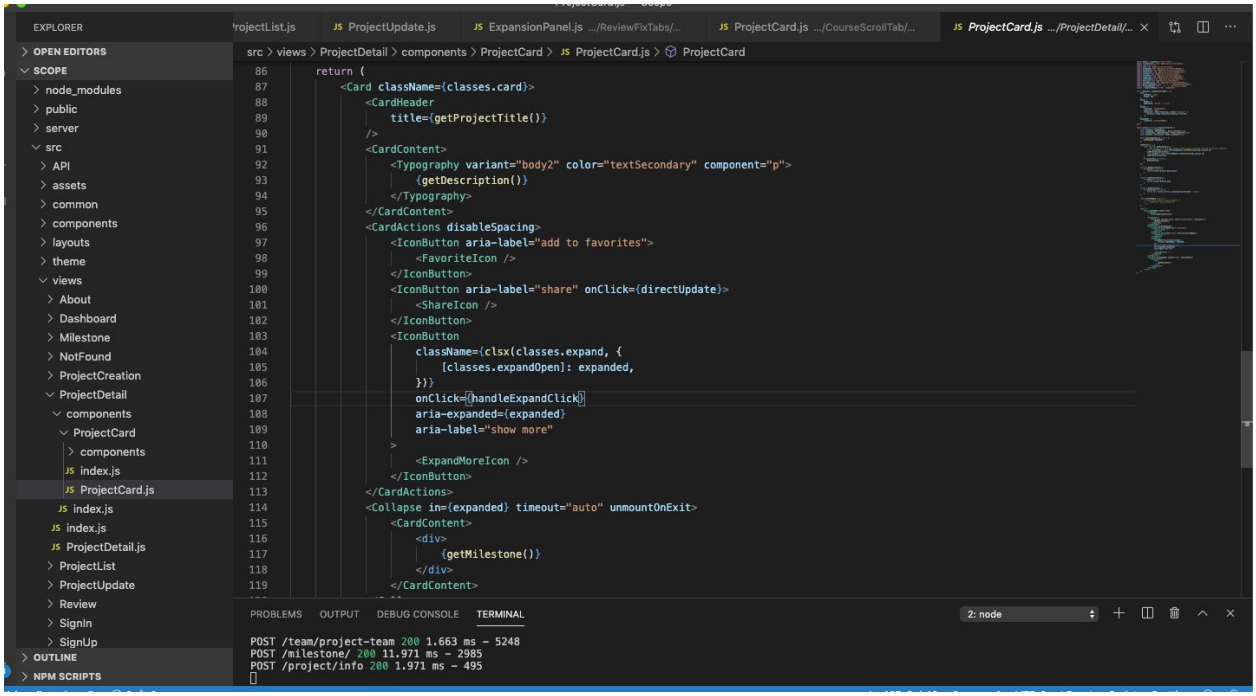
All of the milestones information store in the Milestone table of the scope



ProjectHasMilestones table indicates what and how many milestones are in specific project

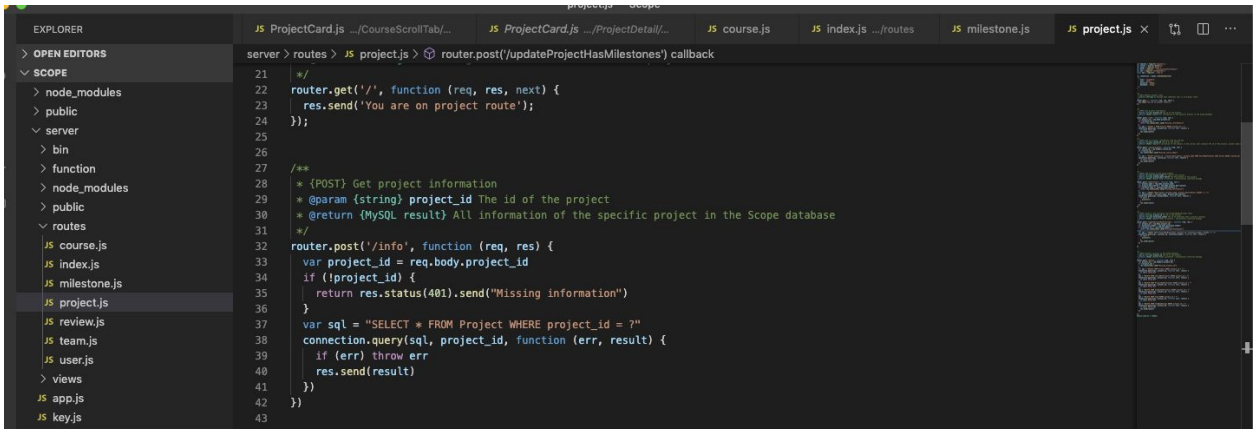
As shown in the screenshot, project ID 1 which is Three Tier Application has three milestones 1, 2 and 3. Milestones 1, 2 and 3 contains information matches the font-end display.

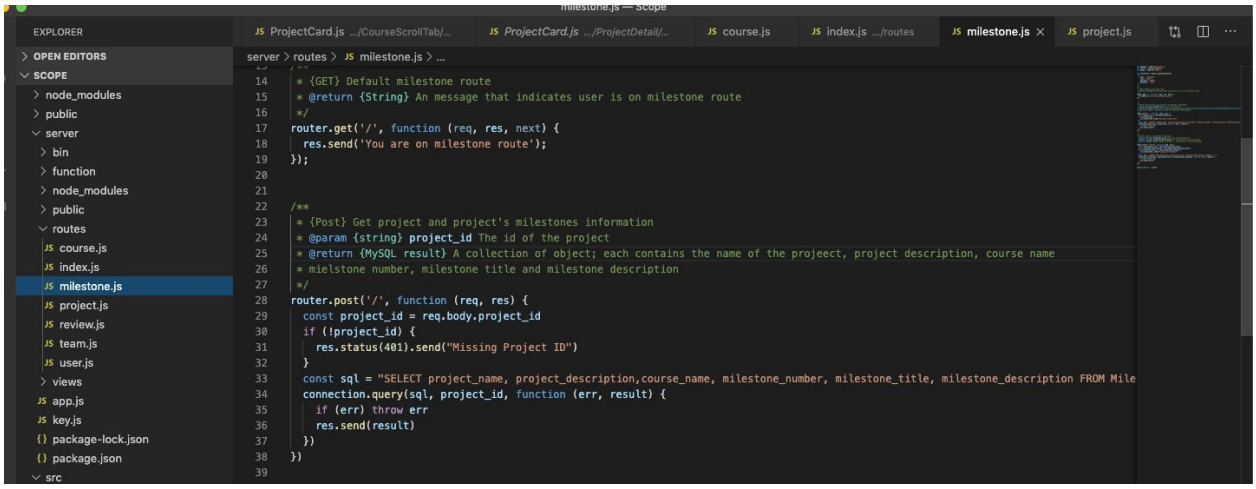
Front-End Code



The above screenshot shows the font-end sources code of Project Card in the ProjectDetails page.

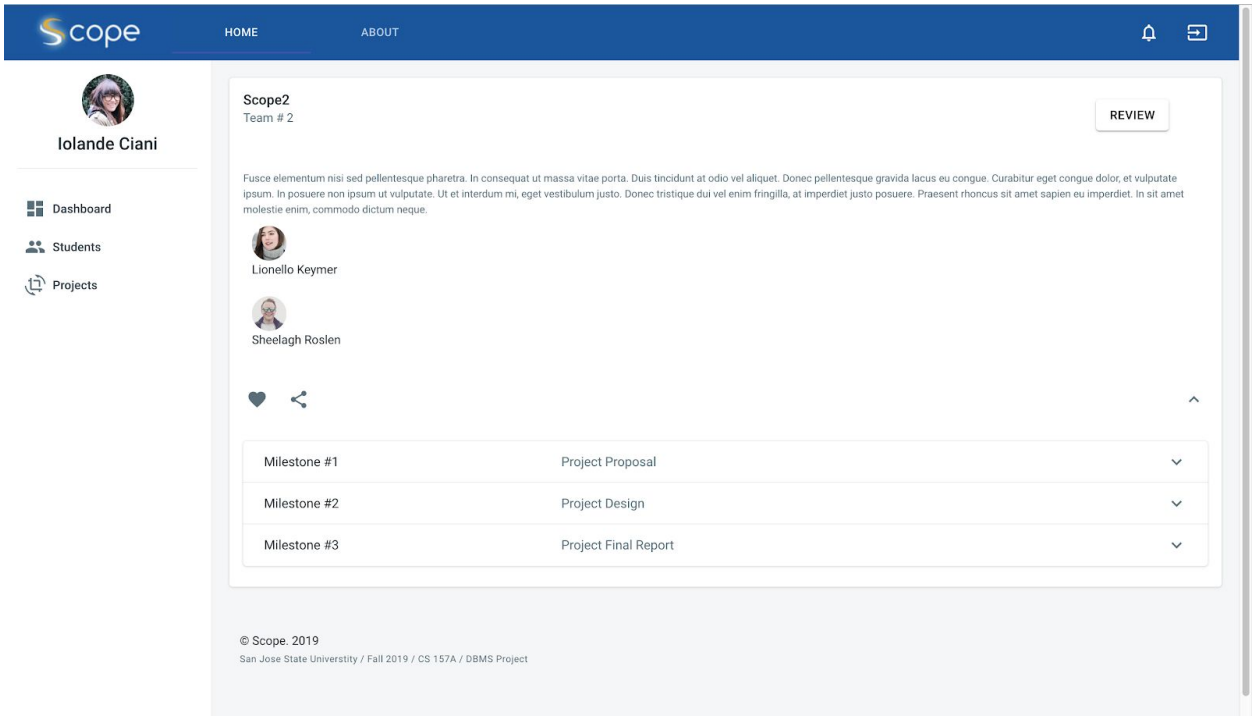
Backend Code





The above screenshots show two API calls to fetch project information and milestone information of the project.

Display Team Information & Display Team Member
Front-End

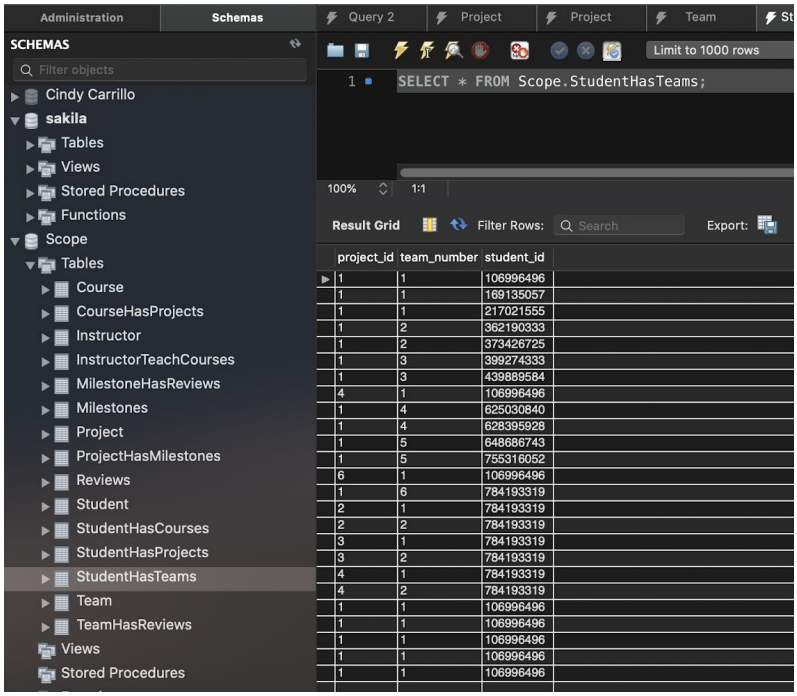


The above screenshot shows the page that appears when the ‘MORE’ button is clicked for a specific team in the dashboard or project page. There is a list of team members for that team and a short title and description, as well as the milestones.

Front-End Code

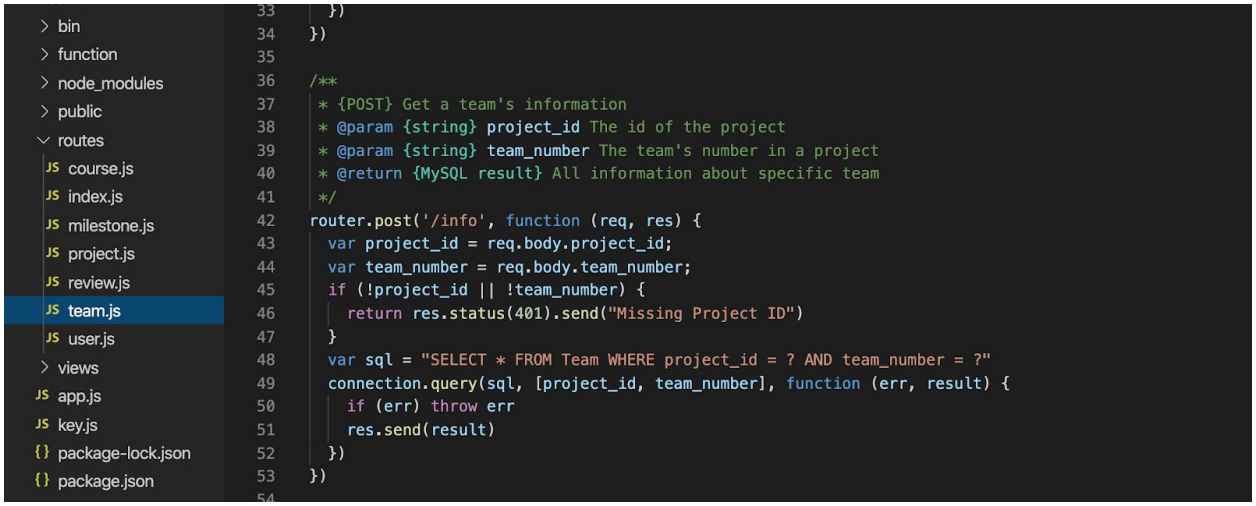
The above screenshot shows the code for displaying team information and team members by calling the getTeamMember() function which just returns a member if there is one assigned to that team

Table



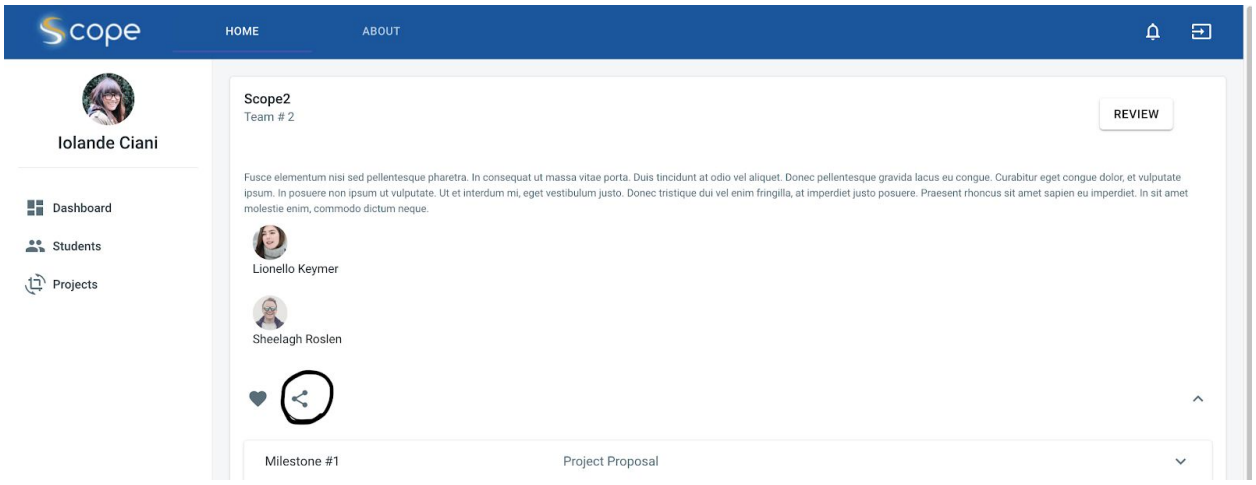
The StudentHasTeams table shows the relationship between student and their teams to display which students are in which team.

Backend Code

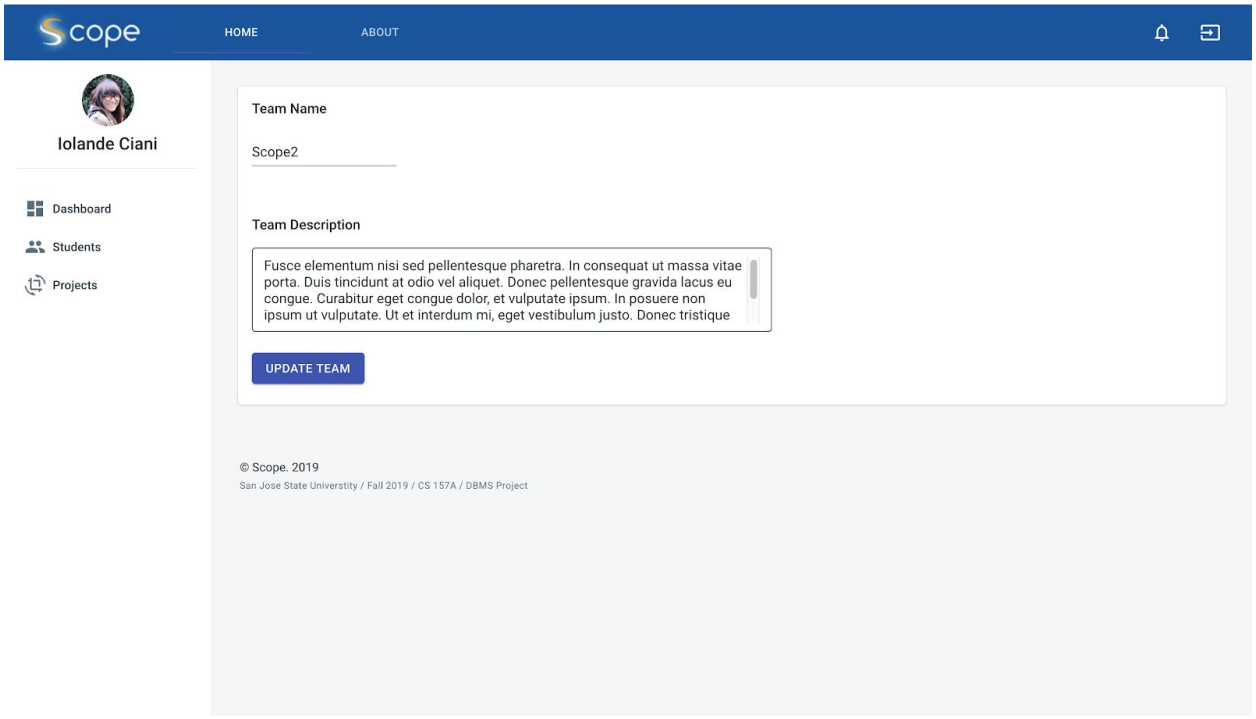


The above screenshot shows how team information is displayed from database to UI.

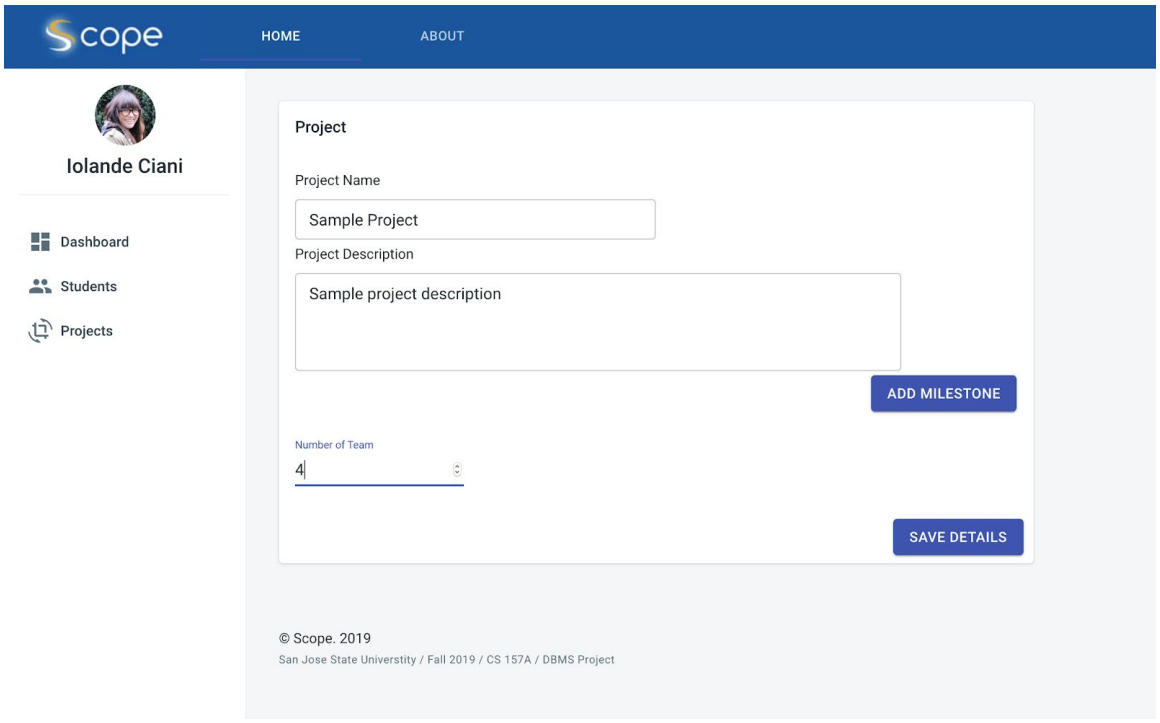
Edit Team & Add Team
Front-End



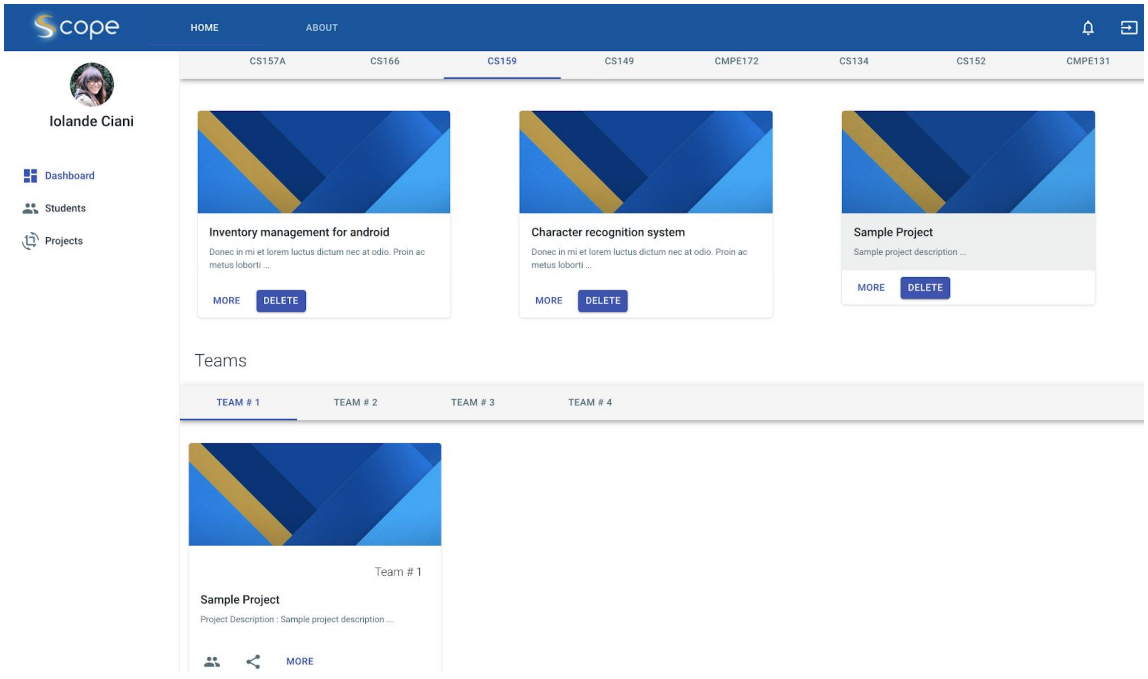
The above screenshot shows the circled icon to edit a team's information.



Once the button is clicked, a new page will appear with the team's information to edit and update.



This screenshot shows when creating a project, the user can specify how many teams the project will have.

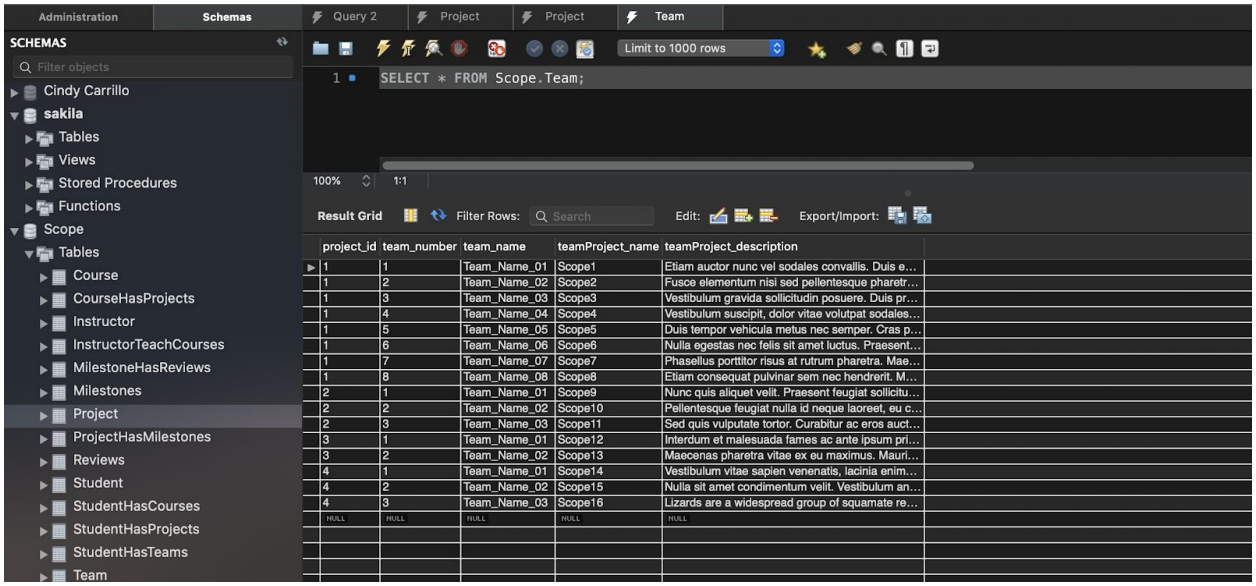


After creating teams, returning to the dashboard page will display the teams for that project.

Front-end Code

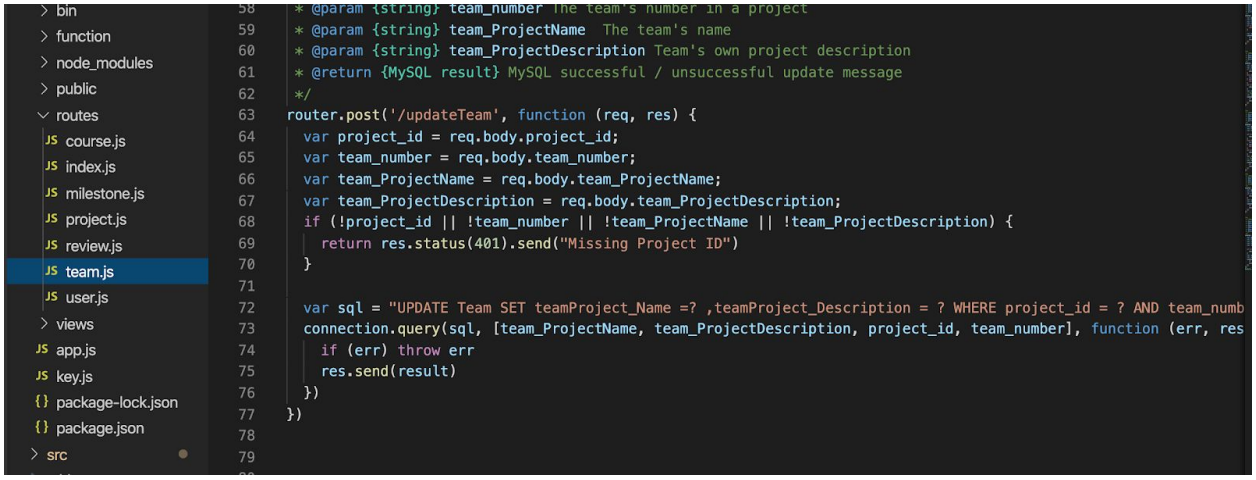
This screenshot shows the function calls to update the team information such as name and description.

Tables



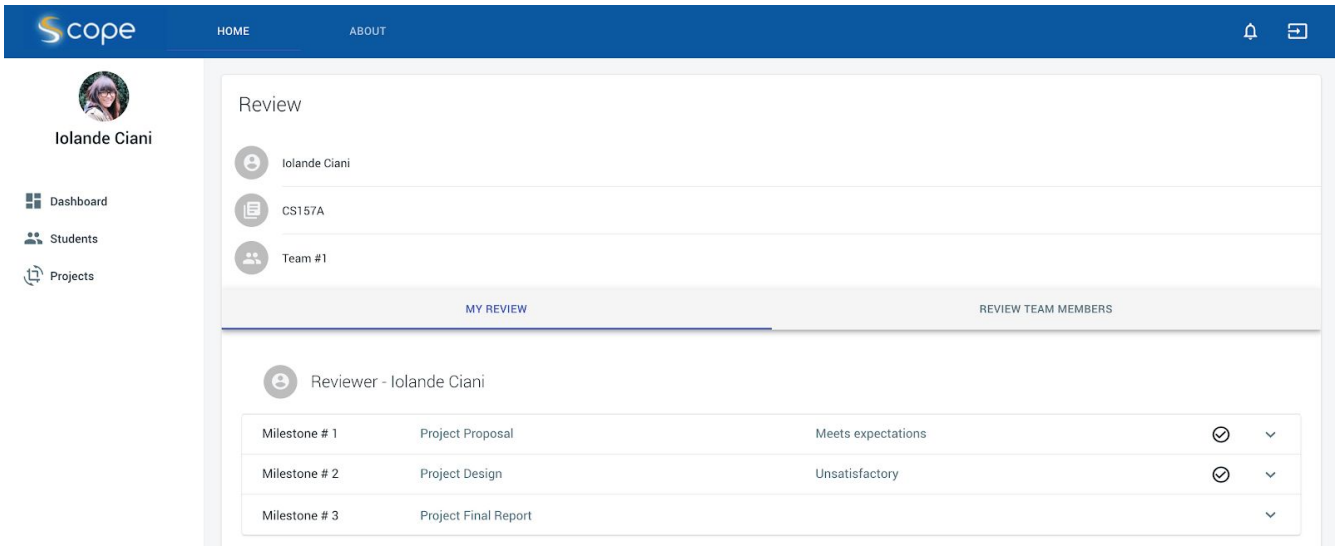
This screenshot shows the Team table which displays the team name, project name, description, and a team_number associated with it. Then a project_id holds the relationship to assign each project a specific number of teams.

Backend Code



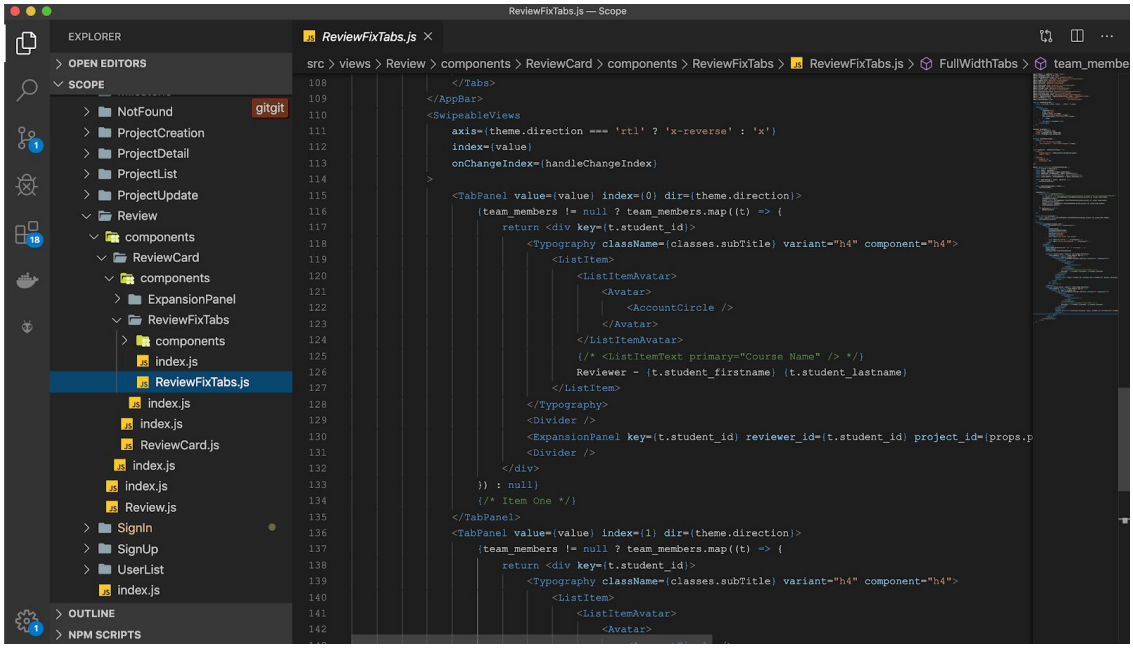
This screenshot shows the backend code that updates the database table once the user updates the team information on Scope.

Add Performance Review
Front-End

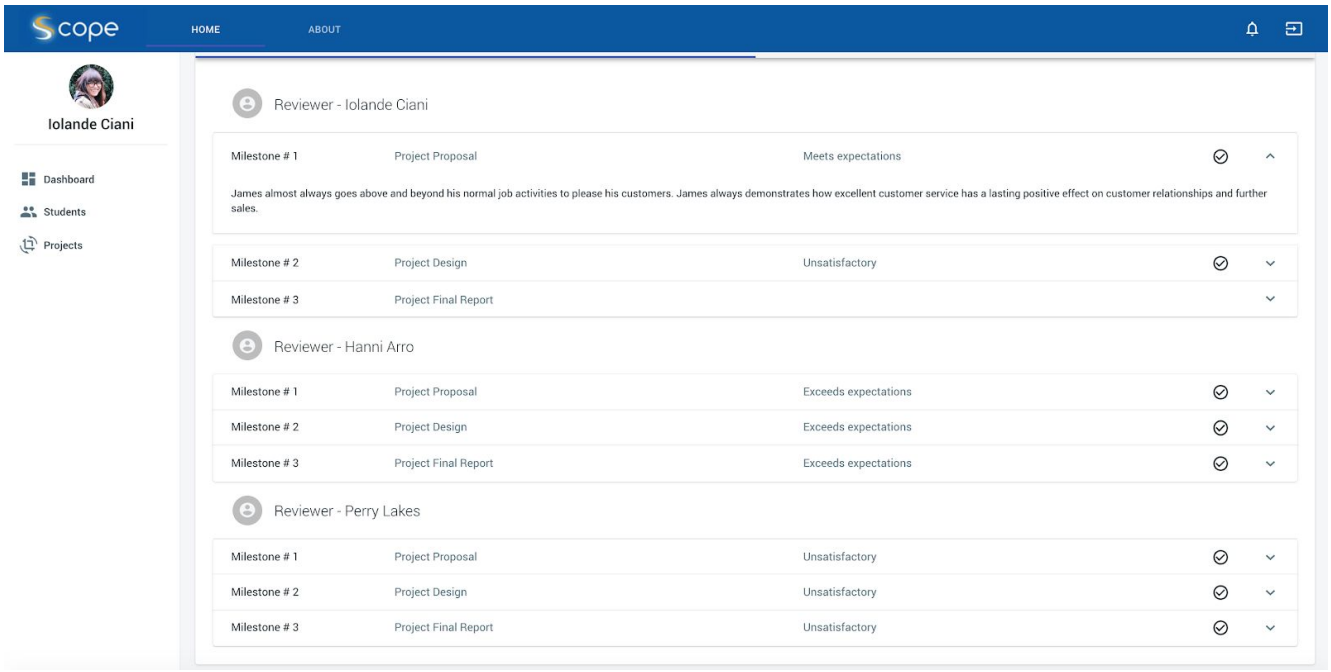


The above Review Page screenshot shows three main components to properly display the student information, the class they are taking and what team member they belong in. Additionally it shows two other navigation panels for the student review and team member the student can review.

Front-End Code

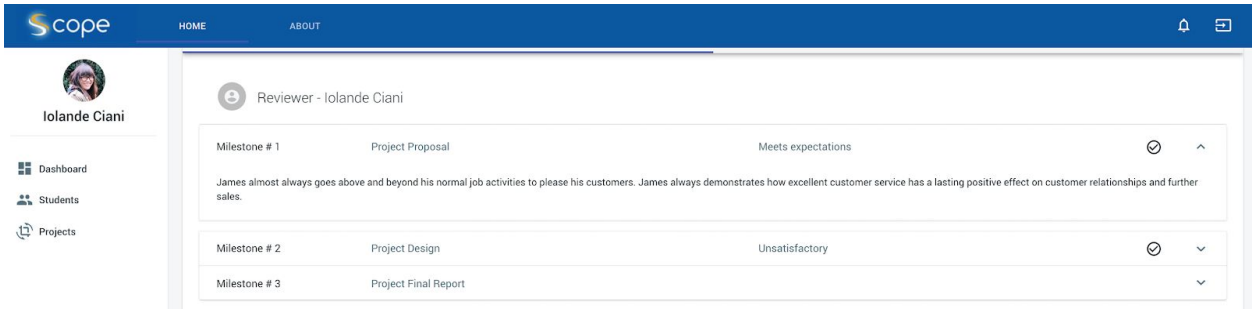


My Review Section



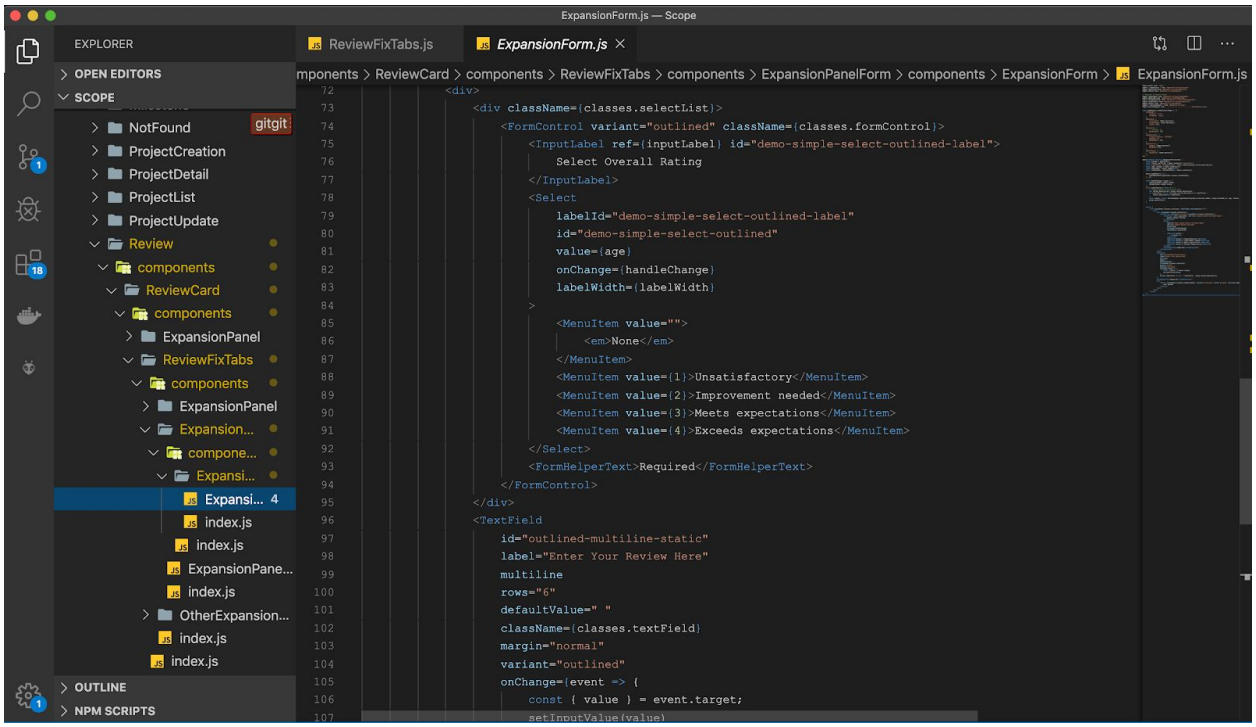
The above My Review Section screenshot shows all the student reviewer information with all the millstone information with dropdown detailed information of that specific milestone performance reviews the student received from their team members.

Self Assessment
Reviewer - Current User

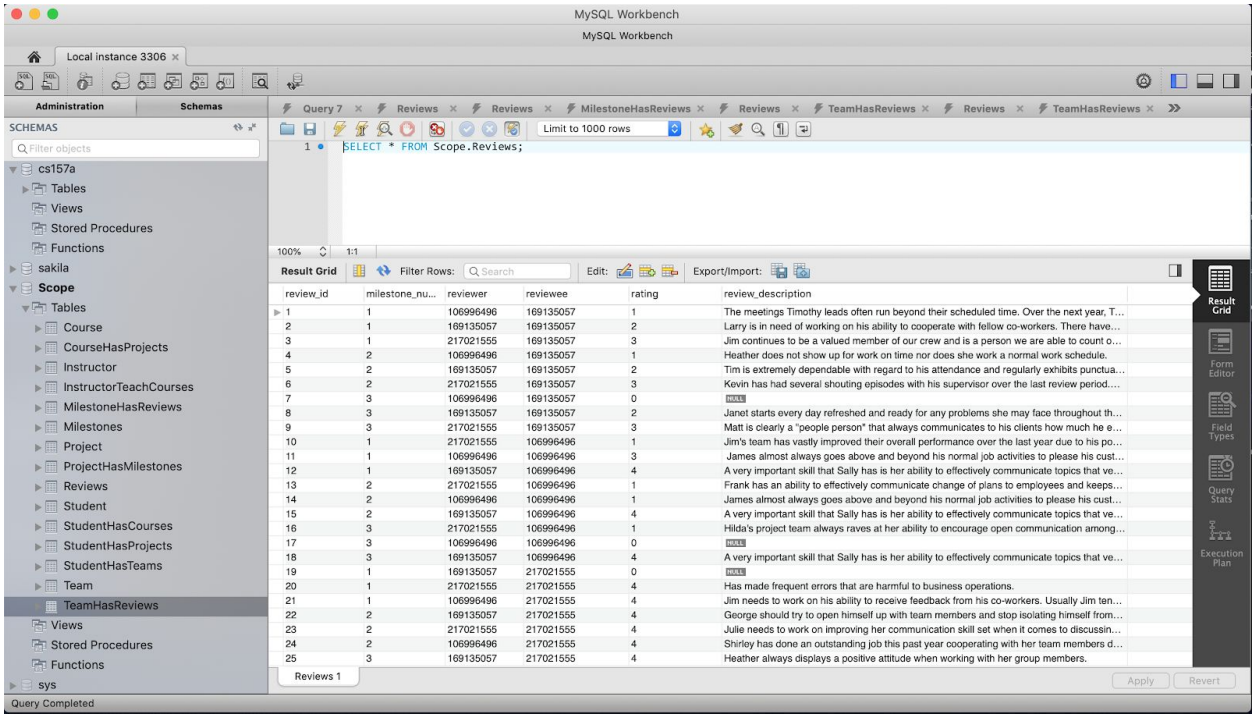


The above screenshot section shows where the current logged in student will be able to click the dropdown section and get a chance to write a self assessment of each milestone of the project.

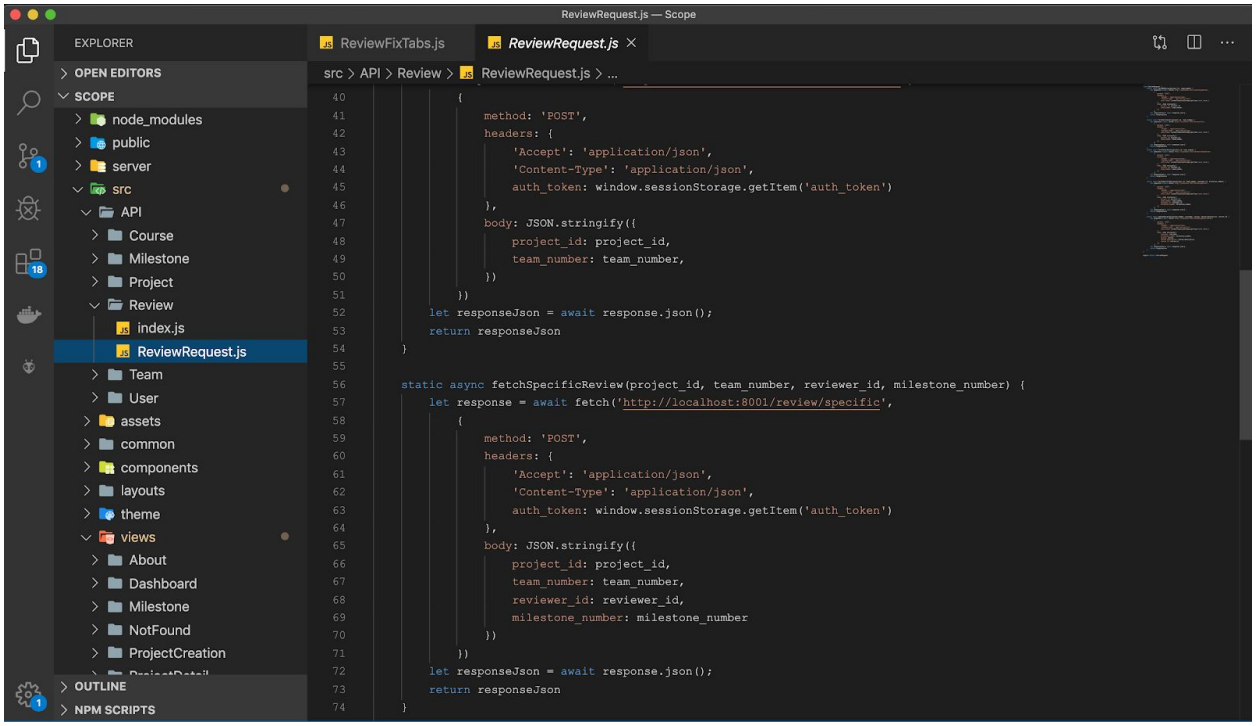
Front-End Code



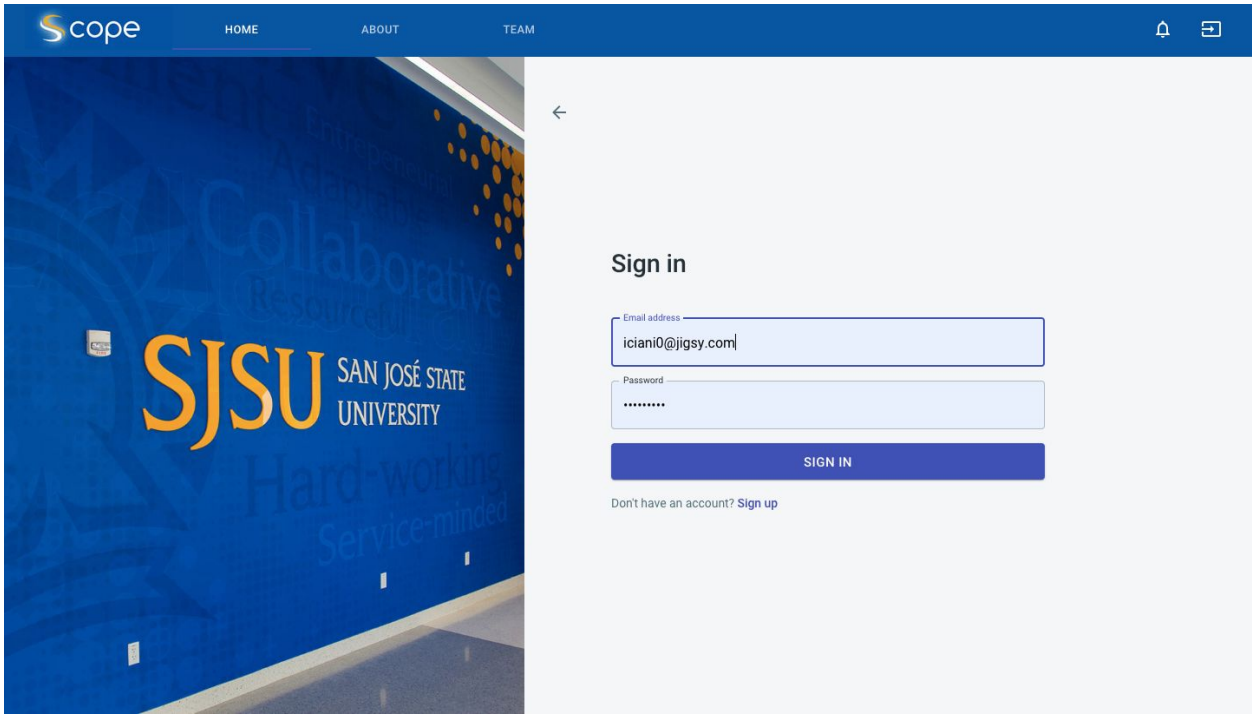
Tables



Backend Code

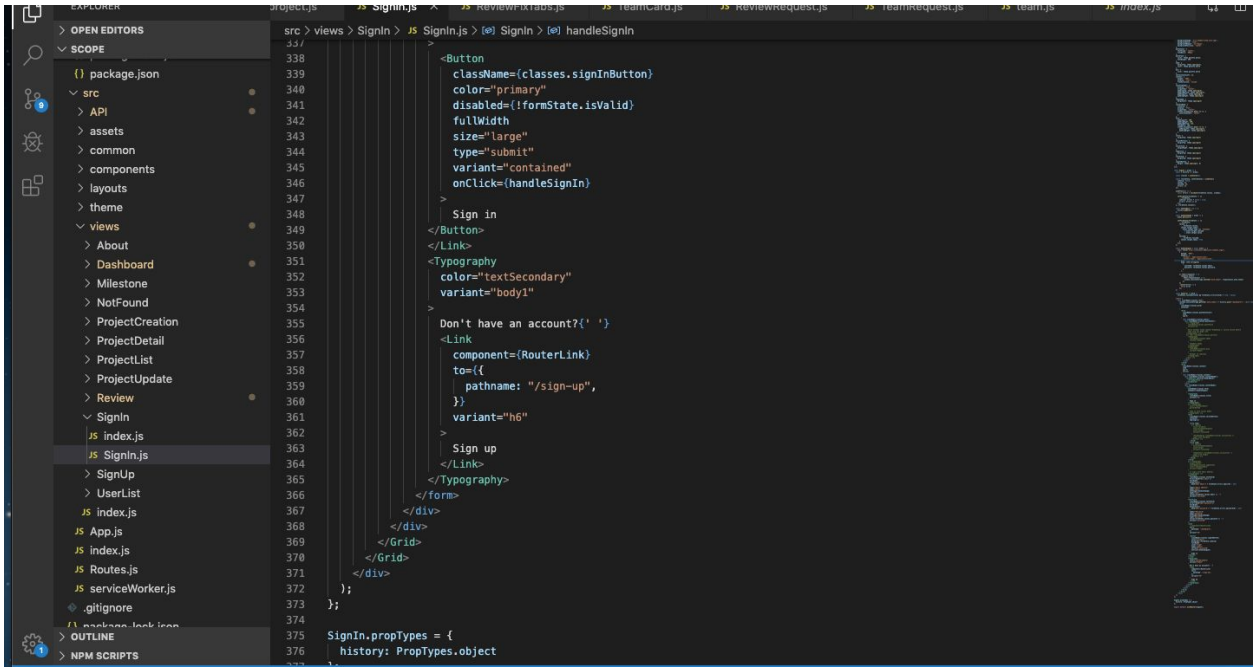


Student Sign in
Front-End



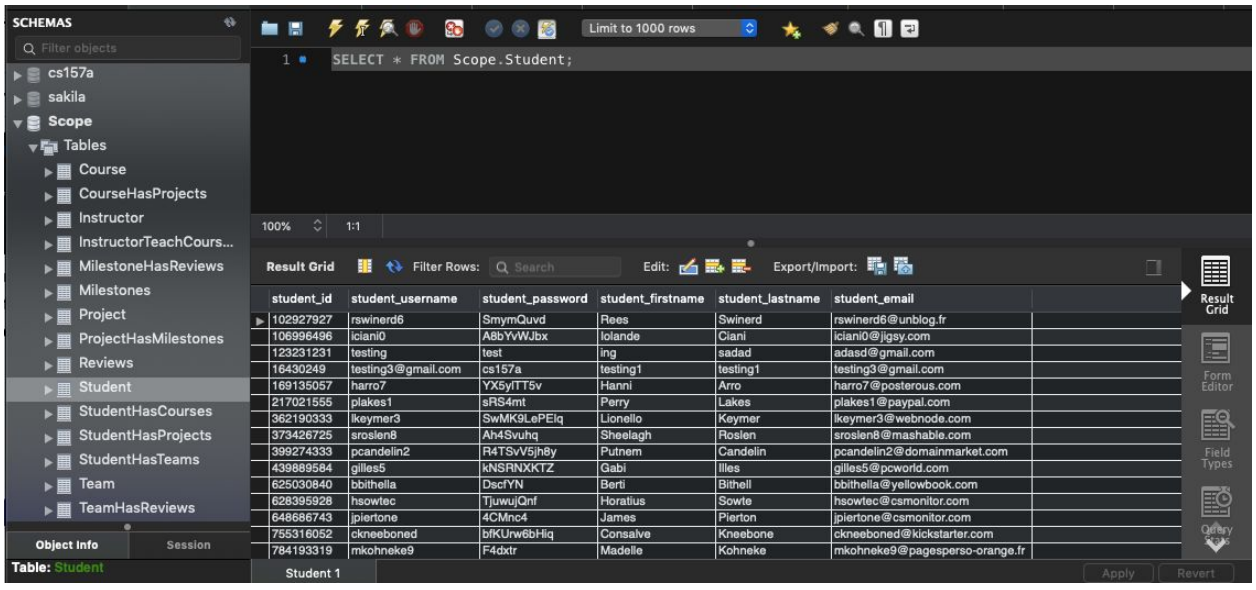
The above screenshot shows the user sign-in page. The email input form contains input validation to ensure the user enters a valid email address.

Front-End Code



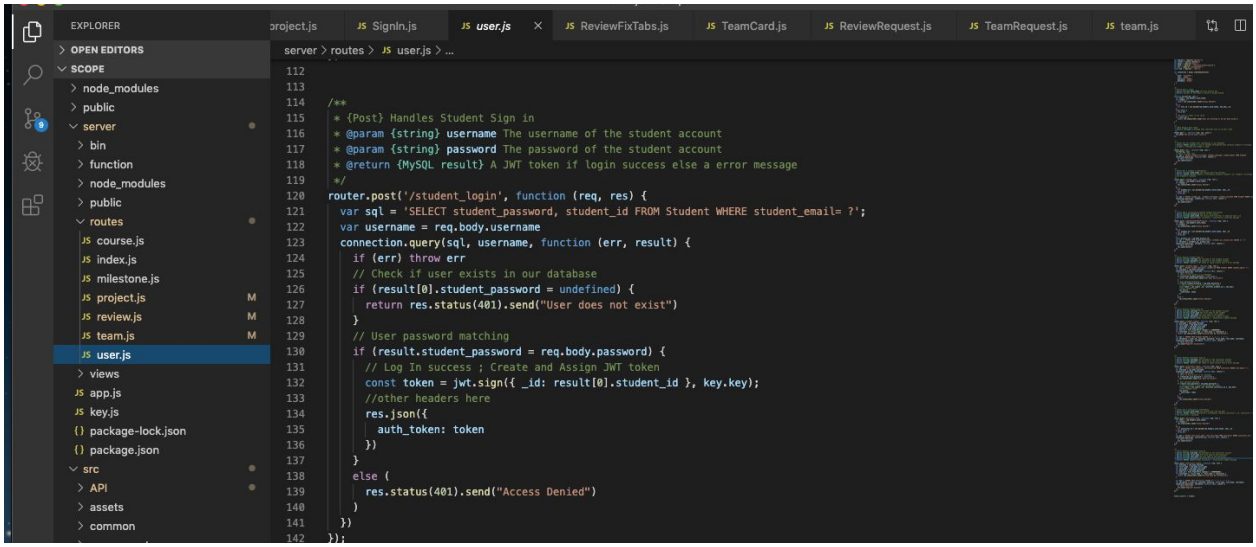
The above screenshot shows the corresponding font-end sign in source code.

Table



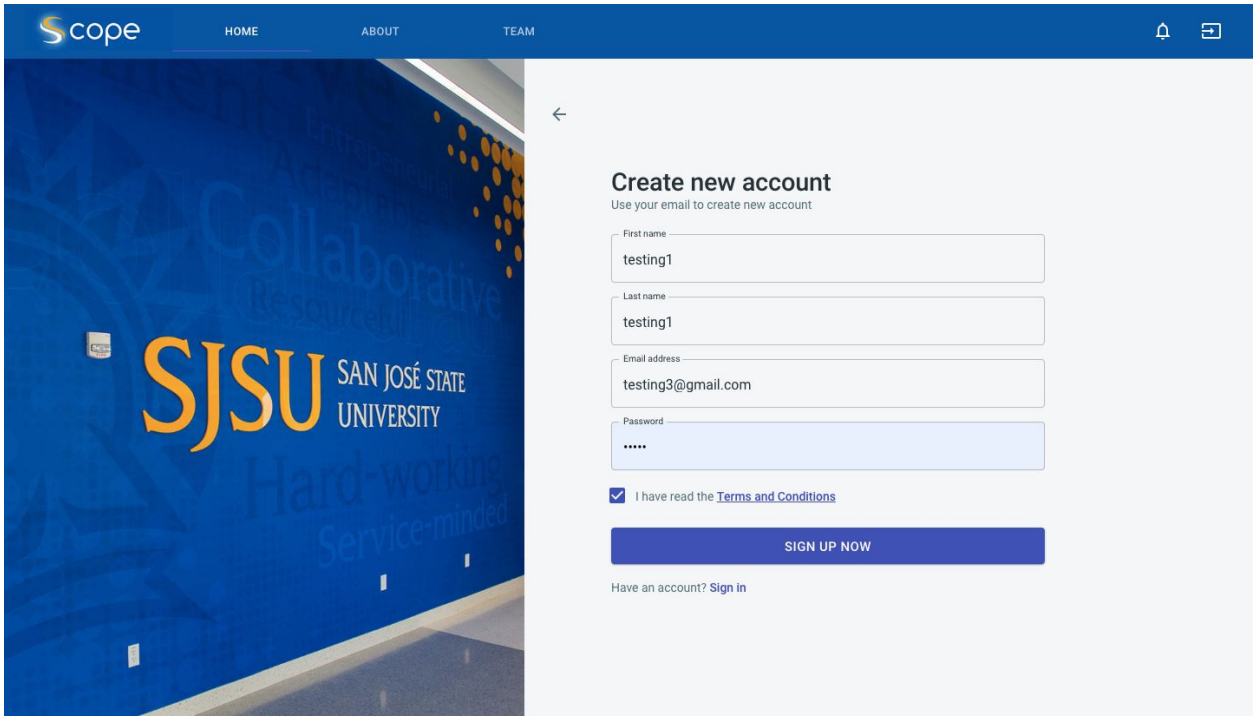
Student table stores all the information and credential for the student user. The User sign-in page will compare and validates the email and password inputs to the data stored in the Student table.

Backend Code



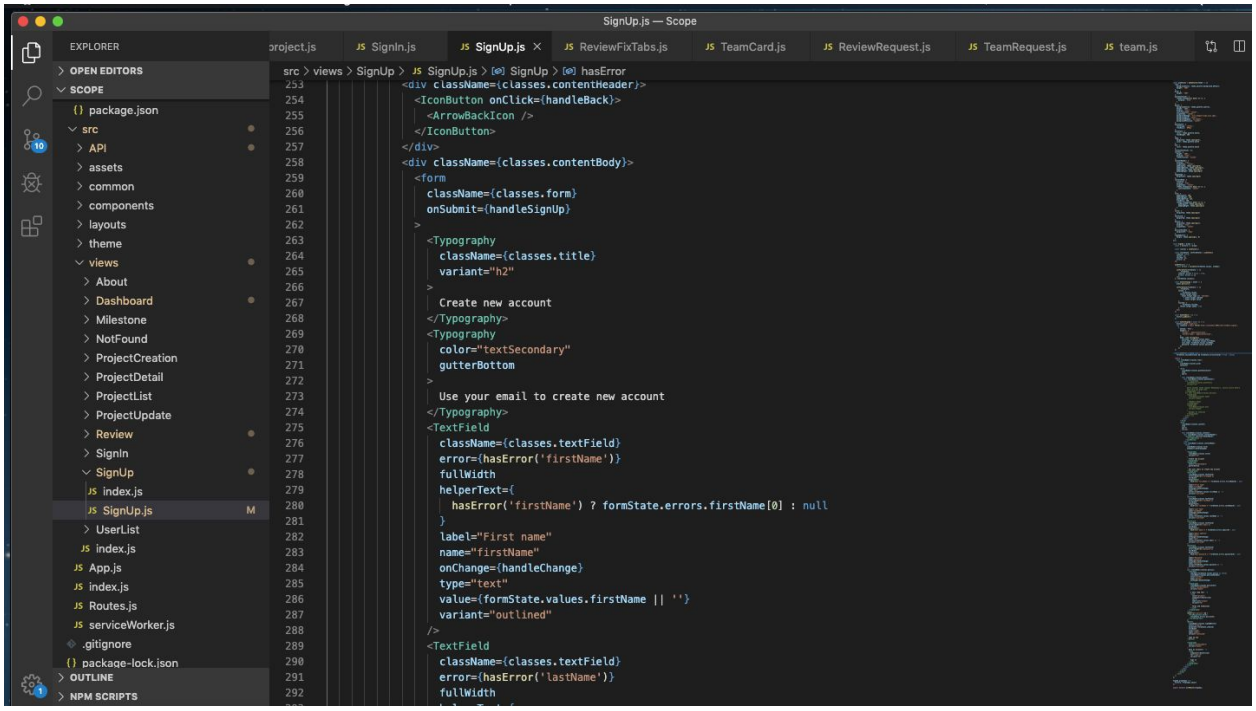
The above screenshot shows the corresponding back-end API for user authentication. When the user successfully logs in using their credentials, the server will send back a JSON Web Token as the session key.

Student Sign Up
Front-End



The above screenshot shows the user sign-up page. The user of Scope will be asking for their first name, last name, email address, and password for the user registration.

Front-End Code



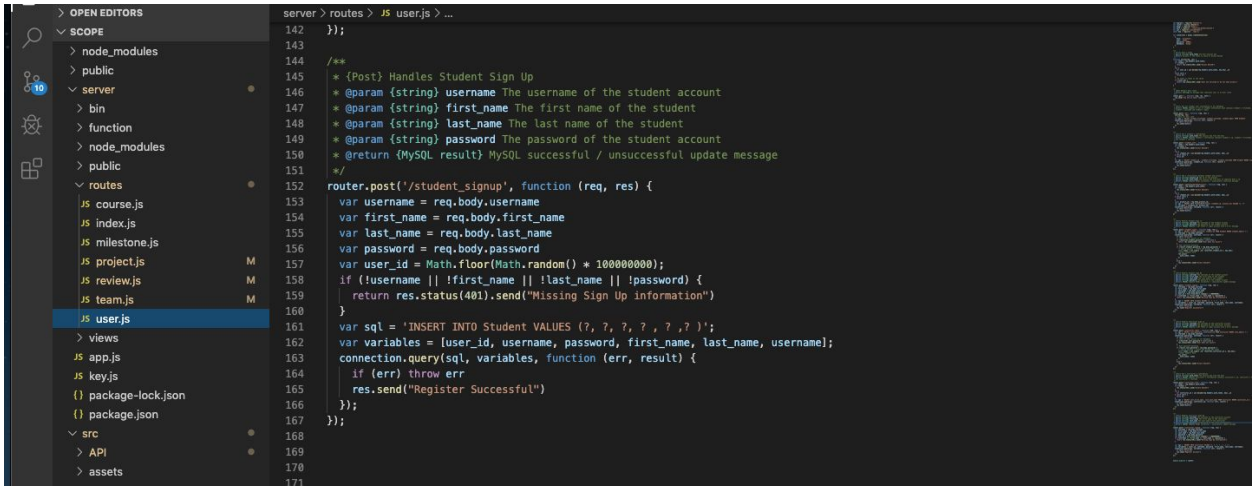
The above screenshot shows the corresponding font-end sign up page source code.

Table

student_id	student_username	student_password	student_firstname	student_lastname	student_email
102927927	rswinerd6	SmymQuvd	Rees	Swinerd	rswinerd6@unblog.fr
106996496	iciani0	A8bYvWJbx	Iolande	Ciani	iciani0@jigsy.com
123231231	testing	test	ing	sadad	adasd@gmail.com
16430249	testing3@gmail.com	cs157a	testing1	testing1	testing3@gmail.com
169135057	harro7	YX5yITT5v	Hanni	Arro	harro7@posterous.com
217021555	plakes1	sRS4mt	Perry	Lakes	plakes1@paypal.com
362190333	lkeymer3	SwMK9LePEIq	Lionello	Keymer	lkeymer3@webnode.com
373426725	sroslen8	Ah4Svuhq	Sheelagh	Roslen	sroslen8@mashable.com
399274333	pcandelin2	R4TSvV5jh8y	Putnem	Candelin	pcandelin2@domainmarket.com
439889584	gilles5	kNSRNXKTZ	Gabi	Illes	gilles5@pcworld.com
625030840	bbithella	DscfYN	Berti	Bithell	bbithella@yellowbook.com
628395928	hsowtec	TjuwujQnf	Horatius	Sowte	hsowtec@csmonitor.com
648686743	jpiertone	4CMnc4	James	Pierton	jpiertone@csmonitor.com
755316052	ckneeboned	bfKUrw6bHiq	Consalve	Kneebone	ckneeboned@kickstarter.com
784193319	mkohneke9	F4dxtr	Madelle	Kohneke	mkohneke9@pagesperso-orange.fr

After the user successfully signs up an account. A new tuple instance will be stored in the Student Table.

Backend Code



The above screenshot shows the corresponding back-end API for user sign up. When the user successfully sign up an account, the user's information will be stored in the Student Table.

Normalization : BCNF

Course

Course (course_id, course_name, course_section, start_date, end_date)

Let A = course_id, B = course_name, C= course_section, D=start_date, E=end_date

Functional Dependency: A-> BCDE

A-> BCDE is a nontrivial functional dependency and A is a superkey; therefore, the Course table is in BCNF

Project

Project (project_id, project_name, project_description)

Let A = project_id, B= project_name, C=project_description

Functional Dependency: A-> BC

A-> BC is a nontrivial functional dependency and A is a superkey; therefore, the Project table is in BCNF

CourseHasProjects

CourseHasProjects is a table that establishes the relationship between Course and Projects table. The CourseHasProjects table takes two primary keys course_id and project_id from Course table and Project table respectively. Since CourseHasProjects is a relationship table, it is already in BCNF.

Instructor

Instructor (instructor_id, inst_user_name, inst_first_name, inst_last_name, inst_email, inst_password)

Let A= instructor_id, B=inst_user_name, C=inst_first_name, D=inst_last_name, E=inst_email, F=inst_password

Functional Dependency: A-> BDCEF

A-> BCDEF is a nontrivial functional dependency and A is a superkey; therefore, the Instructor table is in BCNF

InstructorTeachCourses

InstructorTeachCourses is a table that establishes the relationship between Instructor and Courses. The InstructorTeachCourses table takes two primary keys instuctor_id and course_id from Course table and Instructor table respectively. Since InstructorTeachCourses is a relationship table, it is already in BCNF.

Milestones

Milestones(milestone_number, milestone_title, milestone_description)
Let A = milestone_number, B= milestone_title, and C= milestone_description
Functional Dependency: A -> BC
A-> BC is a nontrivial Functional Dependency and A is a superkey; therefore,
the Review table is in BCNF

Review

Review(review_id, milestone_number, reviewer, reviewee, rating, review_description)
Let A = review_id, B = milestone_number, C = reviewer, D = reviewee, E = rating, F = review_description)
Functional Dependency: A-> BCDEF
A-> BCDEF is a nontrivial Functional Dependency and A is a superkey; therefore,
the Review table is in BCNF

MilestoneHasReviews

MilestoneHasReviews is a table that establishes the relationship between Milestone and Reviews. The MilestoneHasReviews table takes two primary keys milestone_number and review_id from Milestone table and Reviews table respectively. Since MilestoneHasReviews is a relationship table, it is already in BCNF.

ProjectHasMilestones

ProjectHasMilestone is a table that establishes the relationship between Project and Milestone. The ProjectHasMilestone table takes two primary keys project_id and milestone_number from Project table and Milestone table respectively. Since ProjectHasMilestone is a relationship table, it is already in BCNF.

Student

Student (student_id, student_username, student_password, student_firstname, student_lastname, student_email)

Let A= student_id, B= student_username, C=student_password, D=student_firstname, E=student_lastname, F= student_email

Functional Dependency: A-> BDCEF

A-> BCDEF is a nontrivial functional dependency and A is a superkey; therefore, the Student table is in BCNF

StudentHasCourses

StudentHasCourses is a table that establishes the relationship between Student and Courses. The StudentHasCourses table takes two primary keys student_id and course_id from Student table and Course table respectively. Since StudentHasCourses is a relationship table, it is already in BCNF.

StudentHasProjects

StudentHasProjects is a table that establishes the relationship between Student and Project. The StudentHasProjects table takes two primary keys student_id and project_id from Student table and Project table respectively. Since StudentHasProjects is a relationship table, it is already in BCNF.

Team

Team (project_id, team_number, team_name, teamProject_name, teamProject_description)

Let A = project_id, B = team_number, C = team_name, D = teamProject_name, E = teamProject_description

Functional Dependency: AB-> CDE

AB-> CDE is a nontrivial Functional Dependency and AB is a superkey; therefore, the Team table is in BCNF

StudentHasTeams

StudentHasTeams is a table that establishes the relationship between Student and Team. The StudentHasTeams table takes two primary keys project_id and student_id from Project table and Student table respectively. Since StudentHasTeams is a relationship table, it is already in BCNF.

TeamHasReviews

TeamHasReviews is a table that establishes the relationship between team and Reviews. The TeamHasReviews table takes two primary keys project_id and review_id from Project table and Review table respectively. Since TeamHasReviews is a relationship table, it is already in BCNF.

Project Conclusion

Lessons Learned

Abraham

Implementing a full stack web application that runs on RDBMS is one of the most important skills I learned during the whole phase of the project. Through the project, I believe I have learned some of the most important aspects of software life cycle development process. Throughout the project we have pushed our product to have dynamic, modern, and powerful usebality. From initial brainstorming of the idea, all the way to the design, documenting, development, implementing and testing phases we have tried to follow and apply agile industry standard software development methods.

Throughout the project, I also have learned to use some of the most powerful software development tools. Tools like GitHub for source code repository and sharing, Zenhub for software project management, and Material UI for Front End development. Most importantly, I have learned how to implement 3-Tier application architecture using Node JS + React + Express JS and MySQL combined. I strongly believe that this is one of the most crucial lessons I learned as a Software Engineering Student.

The last but not the least lesson I learned from this project is how to collaborate effectively with team members. One of the daunting tasks when it comes to working with others is the skill gap that might occur during the initial stage of the project, in my personal experience this kind of challenge usually tend to drag the project behind. Even if our team came from different backgrounds and technical skill levels, I am so grateful we were able to overcome our challenges and deliver an amazing fully functional product at the end.

Cindy

Throughout the course of this project, I had the opportunity to develop and sharpen numerous soft and hard skills with my team. From learning how to work with a team effectively to successfully complete the project, to learning new technology that is required to build the application.

I learned what it takes to work effectively with a diverse team where each member brings a unique set of skills, backgrounds, experiences, and ideas to the table. I learned the importance of setting time aside to meet deadlines and meet as a group to ensure everyone was on the same page and no time was wasted.

When it comes to hard skills, I learned how to use software tools such as Zenhub and Github, and how to use and create a 3-tier architecture for our application using React, NodeJS, Express JS, and MySQL. All the software was new to me and I enjoyed having the opportunity to learn challenging new skills throughout this project that would benefit my future career as a software engineer.

Yulong

Through this DB project, I have an opportunity to apply MySQL knowledge taught in class into a real three-tier application. As a group, we learned using Github and Zenhub for Scope's version control and project management. I also learned new frameworks such as React, NodeJS, and Express during the development of Scope. The most important lesson that I learned in this project is not about technical skills or new technology; it's about the collaboration with the team.

Future Improvements

For the future of Scope, our team envisions a set of new features to improve user experience and performance. These features include the ability to add, delete, and edit a course, edit a project once created, and display student history when searching for a student. Below is a detailed list of the requirements:

Add course:

- The user shall be able to add a new course by clicking the add button. After clicking the plus button, the user shall be able to fill in the name, course Id, and course period on the pop-up form.
- The system shall update the database with the new course associated with the instructor and update the user interface with new courses.

Delete course:

- The user can delete a course by clicking the menu option on the bottom of each displayed course.
- The system shall update the database based on the user action.

Edit course:

- The user shall be able to edit a course by clicking the menu option on the top right of each displayed course.
- The system shall display a form for the user to edit course information. The system shall update the database based on the user action.

Edit project:

- The student user has the ability to edit a project by clicking the menu option on the top right of each displayed project.
- The system shall display a form for the user to edit project information and a warning when user trying to delete a project. The system shall update the database based on the user action.

Display Student History:

- The user shall be able to click on the Student component and review student's history on past projects they have been on and their reviews.
- The system shall retrieve the student history projects in the database and display on the dashboard as an individual component.

Overall, for the future of Scope, not only do we hope to improve user experience and performance, but also safety within the app since we hope universities will incorporate Scope into their software tools to help close the gap between transparency, growth, and improvement in students and their project work.