

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 usher a new opportunity for transparency, improvement, and growth among students.

The Problem

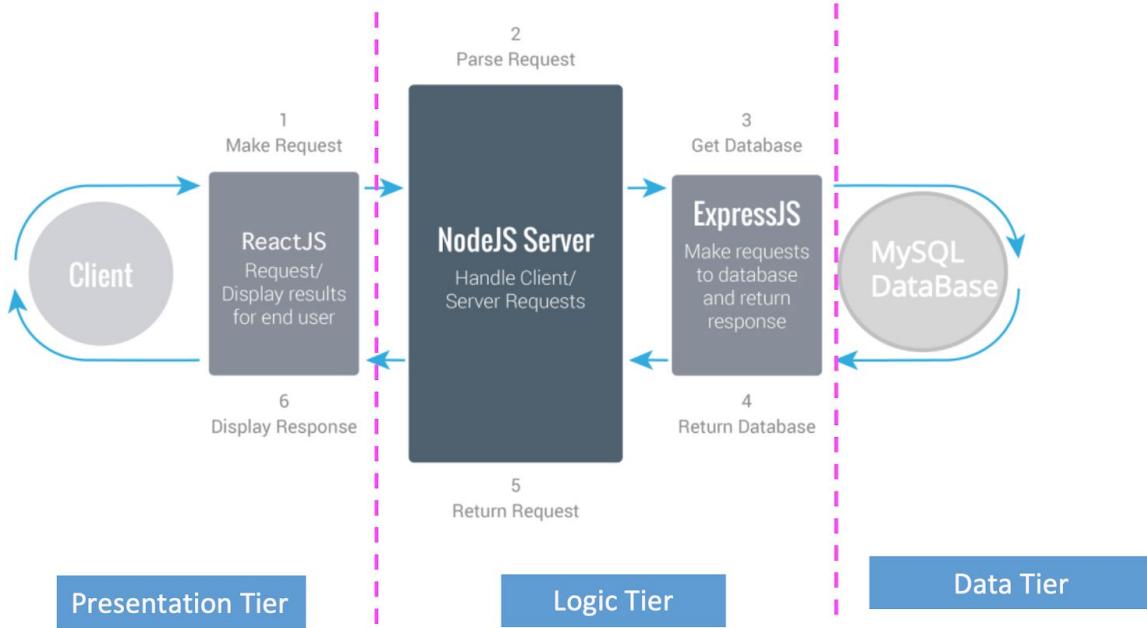
As technology leaps into the next level, the way education is delivered and measured is also rapidly evolving. The traditional way of grading and testing to measure student skills is slowly fading and upgrading with more concrete and skilled based projects that are meant to make students ready for the '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 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.

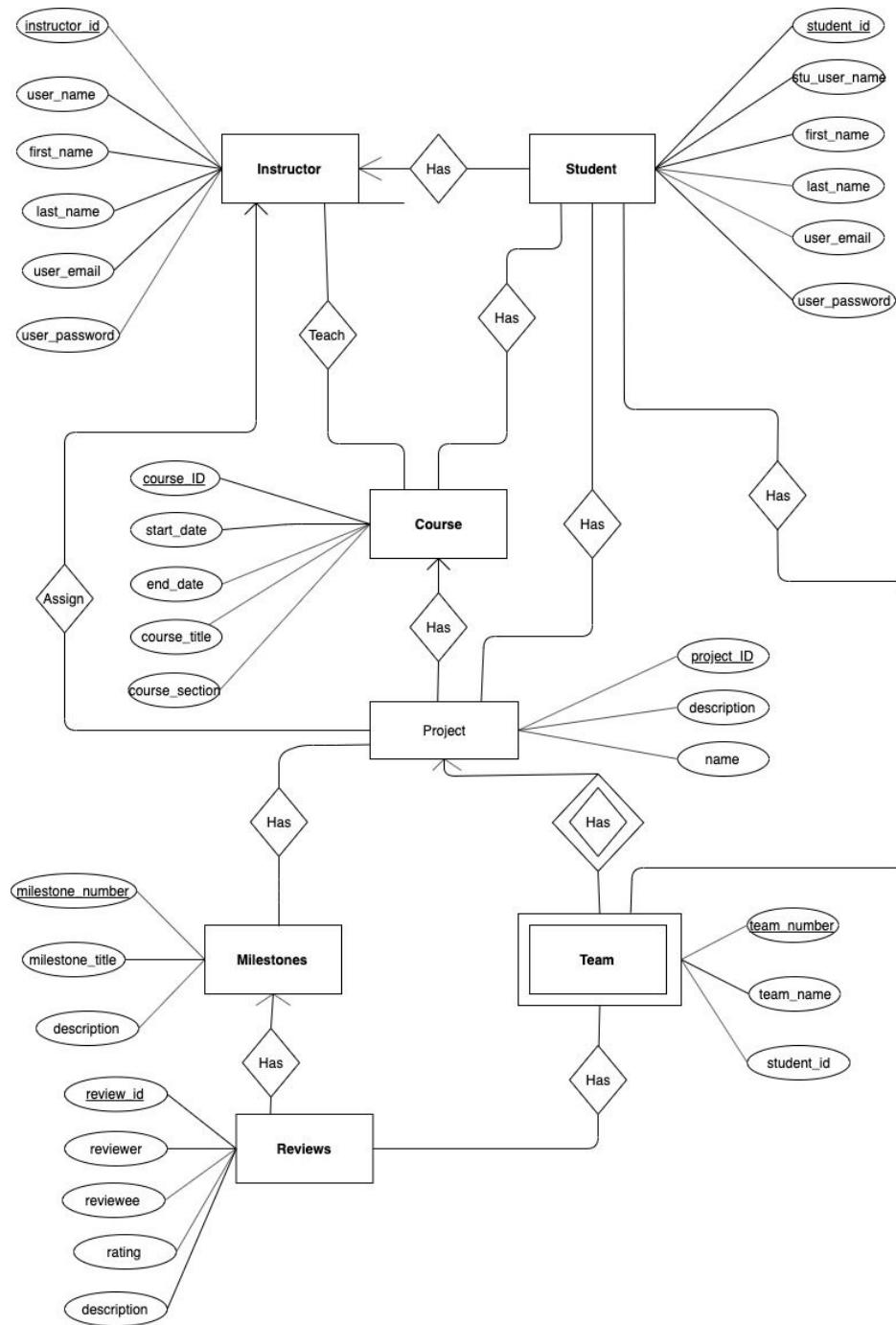
Interface

- 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



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 milestone_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_member, 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

```
Abraham — mysql -u cs157a -p — 172x38

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 | mochterlonie@telegraph.co.uk | NOgnKTfu |
| 0689609795 | hdyment2 | Hewet | Dyment | hdyment2@blinklist.com | J9jv6ydkICfj |
| 1772592293 | mjerredb | Maxie | Jerred | mjerredb@stanford.edu | zvTaAkL |
| 1878210823 | lhasteda | Lynnea | Hasted | lhasteda@xinhuonet.com | xvnSYm |
| 2389592635 | fciliverd4 | Flin | Clilverd | fciliverd4@ucoz.com | 9AoOM7k |
| 309009278 | mochterlonie0 | Maggi | Ochterlonie | mochterlonie@telegraph.co.uk | NOgnKTfu |
| 3547539625 | mstaceyc5 | Morrie | Stacey | mstaceyc5@hexun.com | att7jlpCV |
| 3726186581 | gcheasmanc | Gayler | Cheasman | gcheasmanc@dedecms.com | fvLSuqH5rD |
| 3876648211 | gpfaaff6 | Germain | Pfaff | gpfaaff6@example.com | UStoxF0QaUf |
| 4003367340 | acaddick1 | Augustine | Caddick | acaddick1@quantcast.com | DQKFRK |
| 5169573901 | rgoodboddy3 | Roby | Goodbody | rgoodboddy3@edublogs.org | 074VYSJ0Mz |
| 5455054880 | ariddiforde | Adina | Riddiford | ariddiforde@examiner.com | cQQxKonl |
| 5510108053 | jpavalka9 | Jarvis | Pavelka | jpavalka9@freewebs.com | GOT5Aw |
| 7691982879 | dwcoollacott7 | Dorri | Woollacott | dwcoollacott7@reverbNation.com | 3xtgkqoHwjD |
| 8191688182 | sgerbi18 | Sarge | Gerbi | sgerbi18@mpg.org | BFZnC8Z3r |
+-----+-----+-----+-----+-----+
16 rows in set (0.00 sec)

mysql>
```

Student Table

```
yulongran — mysql -u root -p — 197x48

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 | ABbYwWJbx | Iolande | Ciani | iciani0@jigsy.com |
| 169135057 | harro7 | YX5yLTT5v | Hanni | Arro | harro7@posterous.com |
| 217821555 | plakes1 | skS4mt | Perry | Lakes | plakes1@paypal.com |
| 362190333 | lkeymer3 | SvMK9LePElq | Lionello | Keymer | lkeymer3@webnode.com |
| 373426725 | sroslen8 | Ah4Svuhq | Sheelagh | Roslen | sroslen8@mashable.com |
| 399274333 | pcandelin2 | R4TSvVSjh8y | Putnem | Candelin | pcandelin2@domainmarket.com |
| 439889584 | gilles5 | KNSRNXKTZ | Gabi | Illes | gilles5@pcworld.com |
| 625030840 | bbithella | DscfyN | Berti | Bithell | bbithella@yellowbook.com |
| 628395928 | hsowtec | TjiuwujOnf | Horatius | Sowte | hsowtec@csmonitor.com |
| 648686743 | jpiertone | 4CMnc4 | James | Pierton | jpiertone@csmonitor.com |
| 755316052 | ckneeboned | bFKUr6bhIq | Consalve | Kneebone | ckneeboned@kickstarter.com |
| 784193319 | mkohneke9 | F4dxtr | Madelle | Kohneke | mkohneke9@pagesperso-orange.fr |
| 900078257 | smullally4 | izBp3mUURE | Shantee | Mullally | smullally4@mashable.com |
| 922485677 | swhilesb | ib0MjHS1XpB | 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 | inst_frist_name | inst_last_name | inst_email | inst_password |
+-----+-----+-----+-----+-----+
| Instructor | inst_frist_name | inst_last_name | inst_email | inst_password |
| Project | inst_frist_name | inst_last_name | inst_email | inst_password |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

2 rows in set (0.01 sec)

mysql> SELECT * FROM Scope.Instructor;
+-----+-----+-----+-----+-----+
| project_id | project_name | project_description | inst_frist_name | inst_last_name | inst_email | inst_password |
+-----+-----+-----+-----+-----+-----+
| 1 | Title 1 | Laceration with foreign body of unspecified front wall of thorax without penetration into thoracic cavity, subsequent encounter | J9j+6ydkIC1 | zvTaAKL | xwv5Ym | 7QcQO |
| 2 | Title 2 | Poisoning by unspecified antiepileptic and sedative-hypnotic drugs, intentional self-harm, sequela | acaddick1@quantcast.com | fVLsugH5xD | UStoXPFQAFP | DOKFRK |
| 3 | Title 3 | Adhesions due to foreign body accidentally left in body following other procedure, initial encounter | spaffff6@example.com | 07AVVY3QMs | 3xtgkx0hWjD | BEZnC8Z3r |
| 4 | Title 4 | Other specified injury of unspecified blood vessel at ankle and foot level, unspecified leg | Goodbodyt | xgoodbody43@duh1.org | G0T3AV | 3xtgkx0hWjD |
| 5 | Title 5 | Pedestrian on foot injured in collision with two- or three-wheeled motor vehicle in traffic accident, subsequent encounter | dwoollactt@everbnation.com | sgerbil88@gmail.org | BEZnC8Z3r | 3xtgkx0hWjD |
| 6 | Title 6 | Malnutrition in the puerperium | att7j1pcV | 3xtgkx0hWjD | BEZnC8Z3r | 3xtgkx0hWjD |
| 7 | Title 7 | Toxic effect of arsenic and its compounds, accidental (unintentional), initial encounter | fVLsugH5xD | UStoXPFQAFP | DOKFRK | BEZnC8Z3r |
| 8 | Title 8 | Osteonecrosis due to drugs, left toe(s) | spaffff6@example.com | 07AVVY3QMs | 3xtgkx0hWjD | BEZnC8Z3r |
| 9 | Title 9 | Exposure to flames in controlled fire, not in building or structure | acaddick1@quantcast.com | fVLsugH5xD | UStoXPFQAFP | DOKFRK |
| 10 | Title 10 | Malignant otitis externa, left ear | Goodbodyt | xgoodbody43@duh1.org | G0T3AV | 3xtgkx0hWjD |
| 11 | Title 11 | Other fracture of lower end of unspecified ulna, subsequent encounter for open fracture type I or II with delayed healing | dwoollactt@everbnation.com | sgerbil88@gmail.org | BEZnC8Z3r | 3xtgkx0hWjD |
| 12 | Title 12 | Contusion and laceration of left cerebrum with loss of consciousness of 31 minutes to 59 minutes, initial encounter | G0T3AV | 3xtgkx0hWjD | BEZnC8Z3r | 3xtgkx0hWjD |
| 13 | Title 13 | Toxic effect of other corrosive organic compounds, accidental (unintentional), sequela | dwoollactt@everbnation.com | sgerbil88@gmail.org | BEZnC8Z3r | 3xtgkx0hWjD |
| 14 | Title 14 | Laceration without foreign body of unspecified part of head | Gerbi | sgerbil88@gmail.org | BEZnC8Z3r | 3xtgkx0hWjD |
| 15 | Title 15 | Other chronic osteomyelitis, tibia and fibula | Gerbi | sgerbil88@gmail.org | BEZnC8Z3r | 3xtgkx0hWjD |
+-----+-----+-----+-----+-----+
15 rows in set (0.00 sec)

16 rows in set (0.00 sec)

mysql>
```

Milestones Table

```
bin — ./mysql -u root -p — ./mysql — mysql -u root -p — 116x38

mysql> describe Milestones;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| milestone_number | int(11) | NO | PRI | NULL |       |
| milestone_title | varchar(40) | NO |     | NULL |       |
| milestone_description | text | NO |     | NULL |       |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> SELECT * FROM Scope.Milestones;
+-----+-----+-----+
| milestone_number | milestone_title | milestone_description |
+-----+-----+-----+
| 1 | Milestone_Title_01 | Lorem ipsum dolor sit amet |
| 2 | Milestone_Title_02 | Yosemite Mammoth lake water ipsum dolor sit amet |
| 3 | Milestone_Title_03 | delivery food water cheese |
| 4 | Milestone_Title_04 | homework pen pencil paper turn it in |
| 5 | Milestone_Title_05 | Water words whale washington wow waste |
| 6 | Milestone_Title_06 | dolor health sit example electricity sea travel solution waste |
| 7 | Milestone_Title_07 | Apple banana strawberry grape |
| 8 | Milestone_Title_08 | Lorem ipsum dolor saver waiver ocean sit amet |
| 9 | Milestone_Title_09 | Color planet Lorem earth table ipsum dolor sit cash amet |
| 10 | Milestone_Title_10 | dashboard recognized student structure homework |
| 11 | Milestone_Title_11 | Plate earthquake |
| 12 | Milestone_Title_12 | Lore dolor sit amet |
| 13 | Milestone_Title_13 | Lorem |
| 14 | Milestone_Title_14 | ipsum dolor |
| 15 | Milestone_Title_15 | dolor amet researcher |
+-----+-----+-----+
15 rows in set (0.00 sec)

mysql>
mysql>
```

Team Table

```
Abraham — mysql -u cs157a -p — 172x34

mysql> show tables;
+-----+
| Tables_in_scope |
+-----+
| Instructor |
| Project |
| Team |
+-----+
3 rows in set (0.00 sec)

mysql> SELECT * FROM Scope.Team;
+-----+-----+-----+
| team_number | team_name | student_id |
+-----+-----+-----+
| 1 | Team_Name_01 | 1238200958 |
| 2 | Team_Name_02 | 5229646268 |
| 3 | Team_Name_03 | 0672935694 |
| 4 | Team_Name_04 | 0812124995 |
| 5 | Team_Name_05 | 6553274002 |
| 6 | Team_Name_06 | 4406946845 |
| 7 | Team_Name_07 | 7440428439 |
| 8 | Team_Name_08 | 2707108634 |
| 9 | Team_Name_09 | 1694424553 |
| 10 | Team_Name_10 | 4607787566 |
| 11 | Team_Name_11 | 1240100329 |
| 12 | Team_Name_12 | 7206484131 |
| 13 | Team_Name_13 | 3755384051 |
| 14 | Team_Name_14 | 0516550462 |
| 15 | Team_Name_15 | 0037269925 |
+-----+-----+-----+
15 rows in set (0.00 sec)

mysql>
```

Reviews Table

```
mysql> describe Reviews;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| review_id | varchar(10) | NO | PRI | NULL |       |
| reviewer | varchar(20) | NO |       | NULL |       |
| reviewee | varchar(20) | NO |       | NULL |       |
| rating | int(1) | NO |       | NULL |       |
| review_description | text | NO |       | NULL |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> SELECT * FROM Reviews;
+-----+-----+-----+-----+-----+
| review_id | reviewer | reviewee | rating | review_description |
+-----+-----+-----+-----+-----+
| 1 | Ree | Greedy | 1 | Has made frequent errors that are harmful to business operations. |
| 10 | Levy | Tweedle | 3 | Uses good judgment in solving problems and working with others. |
| 11 | Debora | Chatterton | 2 | Displays occasional negativity when working with others. |
| 12 | Osbert | Feehily | 4 | Courteous and knowledgeable. |
| 13 | Gina | Roscamps | 4 | Demonstrates "team player" behavior views individual success as imperative to group success. |
| 14 | Abbi | Cremin | 5 | Never misses work without prior approval and appropriate notification. |
| 15 | Taffy | Obllein | 3 | Gets annoyed with clients who ask too many questions. |
| 2 | Anna-diana | Kubacek | 2 | The quality of work produced is unacceptable. |
| 3 | Robinson | Kuhne | 3 | Occasionally calls in to work without prior approval, resulting in unscheduled absences. |
| 4 | Wini | Capelin | 4 | Can always be counted on to work overtime when necessary without complaint. |
| 5 | Leone | Sawyer | 5 | All memos, reports, forms and correspondence are completed on time with no errors. |
| 6 | Dallon | Imbrey | 5 | Quantity of work produced is outstanding. |
| 7 | Leon | Stollberger | 4 | Maintains good working relationships with coworkers. |
| 8 | Jermayne | Siney | 3 | Sets priorities and adjusts them as needed when unexpected situations arise. |
| 9 | Colette | Tolputt | 4 | Can always be counted on to work overtime when necessary without complaint. |
+-----+-----+-----+-----+-----+
15 rows in set (0.00 sec)
```

InstructorHasStudent Table

```
mysql> show tables;
+-----+
| Tables_in_scope |
+-----+
| Instructor |
| InstructorAssignProjects |
| InstructorHasStudents |
| Project |
| Team |
+-----+
5 rows in set (0.00 sec)

mysql> describe InstructorHasStudents;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| instructor_id | varchar(10) | NO | PRI | NULL |       |
| student_id | varchar(10) | NO | PRI | NULL |       |
+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> SELECT * FROM Scope.InstructorHasStudents;
+-----+-----+
| instructor_id | student_id |
+-----+-----+
| 0309009278 | 102927927 |
| 0309009278 | 106996496 |
| 0309009278 | 169135057 |
| 0309009278 | 217021555 |
| 0309009278 | 362190333 |
| 0309009278 | 373426725 |
| 0309009278 | 399274333 |
| 0309009278 | 439889584 |
| 0309009278 | 625030840 |
| 0309009278 | 628395928 |
| 0309009278 | 648686743 |
| 0309009278 | 755316052 |
| 0309009278 | 784193319 |
| 0309009278 | 900078257 |
| 0309009278 | 922485677 |
+-----+-----+
15 rows in set (0.00 sec)

mysql>
```

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

```
Abraham — mysql -u cs157a -p — 180x44

mysql> show tables;
+-----+
| Tables_in_scope |
+-----+
| Instructor      |
| InstructorAssignProjects |
| Project         |
| Team            |
+-----+
4 rows in set (0.03 sec)

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

mysql> SELECT * FROM Scope.InstructorAssignProjects;
+-----+-----+
| instructor_id | project_id |
+-----+-----+
| 0114695369   |      14 |
| 0309009278   |       1 |
| 0689609795   |       3 |
| 1772592293   |      12 |
| 1878210823   |      11 |
| 2389592635   |       5 |
| 3547539625   |       6 |
| 3726186581   |      13 |
| 3876648211   |       7 |
| 4003367340   |       2 |
| 5169573901   |       4 |
| 5455054880   |      15 |
| 5510108053   |      10 |
| 7691982879   |       8 |
| 8191688182   |       9 |
+-----+-----+
15 rows in set (0.00 sec)

mysql>
```

StudentHasProjects Table

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

mysql> SELECT * FROM StudentHasProjects;
+-----+-----+
| student_id | project_id |
+-----+-----+
| 102927927  |      1 |
| 106996496  |      1 |
| 169135057  |      1 |
| 217021555  |      1 |
| 362190333  |      1 |
| 373426725  |      1 |
| 399274333  |      1 |
| 439889584  |      1 |
| 625030840  |      1 |
| 628395928  |      1 |
| 648686743  |      1 |
| 755316052  |      1 |
| 784193319  |      1 |
| 900078257  |      1 |
| 922485677  |      1 |
| 102927927  |      2 |
| 106996496  |      2 |
| 169135057  |      2 |
| 217021555  |      2 |
| 362190333  |      2 |
+-----+-----+
```

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 — 116x38

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>
```

ProjectsHasMilestones Table

```
Abraham — mysql -u cs157a -p — 180x44
+-----+
| Tables_in_scope |
+-----+
| Instructor      |
| InstructorAssignProjects |
| InstructorHasStudents |
| Project          |
| ProjectHasMilestones |
| Team             |
+-----+
6 rows in set (0.00 sec)

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

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

mysql>
```

MilestoneHasReviews Table

```
bin — ./mysql -u root -p — ./mysql — mysql -u root -p — 116x38
mysql> describe MilestoneHasReviews;
+-----+-----+-----+-----+-----+
| Field    | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| milestone_number | int(10) | NO  |     | NULL    |       |
| review_id       | int(10) | NO  |     | NULL    |       |
+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> SELECT * FROM MilestoneHasReviews;
+-----+
| milestone_number | review_id |
+-----+
|      1      |      12      |
|      2      |       2      |
|      3      |       4      |
|      4      |       5      |
|      5      |      10      |
|      6      |       9      |
|      7      |      15      |
|      8      |       3      |
|      9      |      11      |
|     10     |      11      |
|     11     |       1      |
|     12     |       4      |
|     13     |       7      |
|     14     |       6      |
|     15     |       8      |
+-----+
15 rows in set (0.00 sec)

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

TeamHasReviews Table

```
bin — ./mysql -u root -p — ./mysql — mysql -u root -p — 116x38

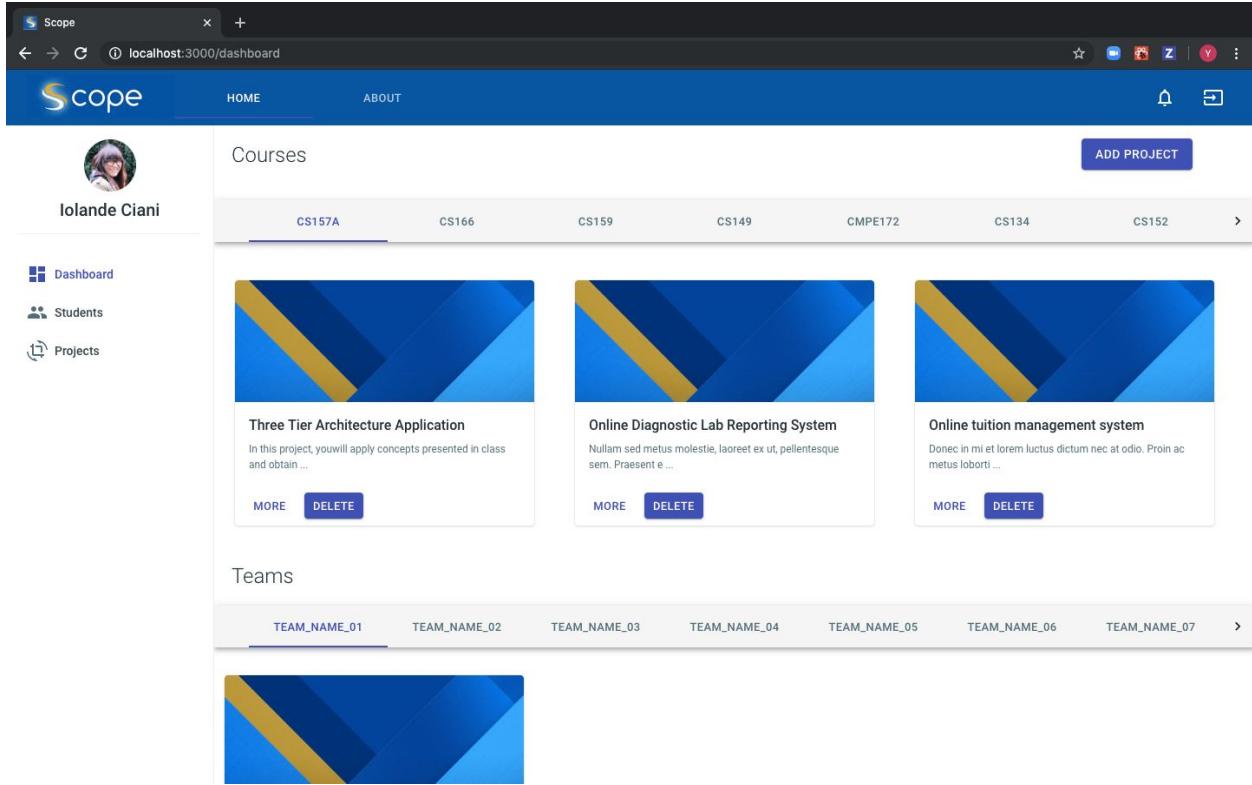
[mysql]> describe TeamHasReviews;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| project_id | int(10) | NO | | NULL | |
| team_number | int(3) | NO | | NULL | |
| review_id | int(10) | NO | | NULL | |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

[mysql]> SELECT * FROM TeamHasReviews;
+-----+-----+-----+
| project_id | team_number | review_id |
+-----+-----+-----+
| 1 | 1 | 573629506 |
| 2 | 2 | 573629324 |
| 3 | 4 | 524229324 |
| 4 | 4 | 224322924 |
| 5 | 5 | 224322224 |
| 6 | 6 | 223422224 |
| 7 | 7 | 234422224 |
| 8 | 8 | 223422224 |
| 9 | 9 | 223477224 |
| 10 | 10 | 226677224 |
| 11 | 11 | 226677624 |
| 12 | 12 | 226674627 |
| 13 | 13 | 226699627 |
| 14 | 14 | 256659627 |
| 15 | 15 | 256669687 |
+-----+-----+-----+
15 rows in set (0.00 sec)

[mysql]>
[mysql]>
[mysql]>
[mysql]>
[mysql]>
```

Implementation

Display Course Font-End



The screenshot shows the Scope dashboard at localhost:3000/dashboard. The interface has a blue header with the Scope logo, a navigation bar with 'HOME' and 'ABOUT' buttons, and a user profile section for 'Iolande Ciani'. On the left, a sidebar includes 'Dashboard', 'Students', and 'Projects'. The main area is titled 'Courses' and lists several projects:

Course	Project Name	Description (Partial)	Actions
CS157A	Three Tier Architecture Application	In this project, you will apply concepts presented in class and obtain ...	MORE DELETE
CS166	Online Diagnostic Lab Reporting System	Nullam sed metus molestie, laoreet ex ut, pellentesque sem. Praesent e ...	MORE DELETE
CS159	Online tuition management system	Donec in mi et lorem luctus dictum nec at odio. Proin ac metus loborti ...	MORE DELETE
CS149			
CMPE172			
CS134			
CS152			

Below the courses, there is a 'Teams' section with a single team entry:

Team Name
TEAM_NAME_01

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

Front-End Code

A screenshot of a code editor (VS Code) showing the file `CourseScrollTab.js`. The editor has a dark theme. On the left is the Explorer sidebar with a tree view of the project structure, including files like `ProjectCard.js`, `Footer.js`, `Sidebar.js`, `TeamCard.js`, `UserList.js`, `Dashboard.js`, and `CourseScrollTab.js`. The main pane shows the `CourseScrollTab.js` code, which contains JSX for an `AppBar` and a `Tabs` component. The code uses `map` functions to generate tabs for each course in the `props.course_list` array. The bottom status bar shows the file path as `CourseScrollTab.js — Scope`, line 85, column 81, and the encoding as UTF-8.

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

Tables

A screenshot of a database management tool (likely pgAdmin) showing the `Course` table in the `Scope` schema. The table has columns: `course_id`, `course_name`, `course_section`, `start_date`, and `end_date`. The data grid shows several rows of course information. Below the table, there is an object info panel for the `Course` table, showing its columns and primary key `course_id`. The columns listed are: `course_id` (varchar(10) PK), `course_name` (varchar(40)), `course_section` (varchar(2)), `start_date` (date), and `end_date` (date).

Course table in the database stored all course information.

The screenshot shows the MySQL Workbench interface. The left sidebar lists databases (cs157a, sakila, Scope) and tables (Course, CourseHasProjects, Instructor, InstructorTeachCourses, MilestoneHasReviews, Milestones, Project, ProjectHasMilestones, Reviews, Student, StudentHasCourses, StudentHasProjects, StudentHasTeams, Team, TeamHasReviews). The main area shows the 'Scope' database with the 'StudentHasCourses' table selected. A query window at the top contains the SQL command: 'SELECT * FROM Scope.StudentHasCourses;'. The results grid shows the following data:

student_id	course_id
102927927	278917254
10696496	278917254
169135057	278917254
217021556	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
10696496	299655012
169135057	299655012
217021556	299655012
362190333	299655012
373426725	299655012
399274333	299655012
439889584	299655012
625030840	299655012
628395928	299655012
648686743	299655012
755316052	299655012

Below the results, the 'Object Info' and 'Session' tabs are visible. The 'Table: StudentHasCourses' tab is selected. It shows the columns: student_id (varchar(10)) and course_id (varchar(10)). The 'Columns:' section lists the same two columns. The 'Object Info' and 'Session' tabs are also shown. At the bottom, the status bar says 'Query Completed'.

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

```

> OPEN EDITORS
  > SCOPE
    > node_modules
      > public
        > server
          > bin
        > function
      > node_modules
        > public
        > routes
          JS course.js
          JS index.js
          JS milestone.js
          JS project.js
          JS review.js
          JS team.js
          JS user.js
        > views
        JS app.js
        JS key.js
        ( package-lock.json
        ( package.json

```

The code editor shows the 'course.js' file under the 'routes' directory. The file contains the following code:

```

server.get('/courses', (req, res) => {
  const studentId = req.query.student_id;
  if (!studentId) {
    return res.status(400).send('Student ID is required');
  }
  const limit = req.query.limit ? parseInt(req.query.limit) : 10;
  const offset = req.query.offset ? parseInt(req.query.offset) : 0;
  const sql = `SELECT * FROM Courses LIMIT ${limit} OFFSET ${offset}`;
  connection.query(sql, (err, result) => {
    if (err) throw err;
    res.send(result);
  });
});

server.post('/student', (req, res) => {
  const auth_token = req.headers['x-auth-token'];
  const decodedToken = jwt.decode(auth_token, key);
  const studentId = decodedToken._id;
  const courseId = req.body.course_id;
  const sql = `INSERT INTO StudentHasCourses (student_id, course_id) VALUES (?, ?)`;
  connection.query(sql, [studentId, courseId], (err, result) => {
    if (err) throw err;
    res.status(201).send('Course added successfully');
  });
});

server.get('/student/:student_id', (req, res) => {
  const studentId = req.params.student_id;
  const limit = req.query.limit ? parseInt(req.query.limit) : 10;
  const offset = req.query.offset ? parseInt(req.query.offset) : 0;
  const sql = `SELECT course_id, course_name FROM Courses JOIN StudentHasCourses ON Courses.id = StudentHasCourses.course_id WHERE StudentHasCourses.student_id = ? LIMIT ${limit} OFFSET ${offset}`;
  connection.query(sql, [studentId], (err, result) => {
    if (err) throw err;
    res.send(result);
  });
});

```

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 screenshot shows the Scope Front-End dashboard. At the top, there is a navigation bar with links for HOME and ABOUT, and icons for notifications and user profile. On the left, a sidebar shows a profile picture of Iolande Ciani and links for Dashboard, Students, and Projects. The main content area is titled 'Courses' and lists several projects:

Project	Description	Actions
CS157A Three Tier Architecture Application	In this project, you will apply concepts presented in class and obtain ...	MORE DELETE
CS166 Online Diagnostic Lab Reporting System	Nullam sed metus molestie, laoreet ex ut, pellentesque sem. Praesent e ...	MORE DELETE
CS159 Online tuition management system	Donec in mi et lorem luctus dictum nec at odio. Proin ac metus loborti ...	MORE DELETE
CS149 CMPE172 CS134 CS152		

Below the courses section is another section titled 'Teams'.

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.

The screenshot shows the Scope application interface. On the left, there's a sidebar with a user profile picture of Iolande Ciani and three navigation options: Dashboard, Students, and Projects. The main content area has a blue header with tabs for HOME and ABOUT. Below the header, there's a section titled 'Courses' with a sub-header 'CS159'. It displays two project cards: 'Inventory management for android' and 'Character recognition system', each with a 'MORE' and 'DELETE' button. To the right of the courses is an 'ADD PROJECT' button. Below the courses is a section titled 'Teams' with a sub-header 'TEAM_NAME_01'. It shows a single team card with a 'More' button.

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

Front-End Code

The screenshot shows a code editor with several tabs open, including 'ProjectCard.js' which is currently active. The code editor displays the following snippet of JavaScript:

```

    ...
    props.onDeleteProject()
  }
  return (
    <Card className={classes.card} onClick={() => {
      props.onChangeProject(props.project.project_id)
    }}>
      <CardMedia
        className={classes.media}
        image="/images/projects/project-card-bg.jpg"
        title="Project Title"
      />
      <CardContent>
        <Typography gutterBottom variant="h5" component="h2">
          {props.project.project_name}
        </Typography>
        <Typography variant="body2" color="textSecondary" component="p">
          {props.project.project_description.slice(0,70) + '...'}
        </Typography>
      </CardContent>
      <CardActionArea>
        <CardActions>
          <Button size="small" color="primary" onClick={directDetail}>
            More
          </Button>
          <Button size="small" color="primary" variant="contained" onClick={deleteProject}>
            Delete
          </Button>
        </CardActions>
      </CardActionArea>
    </Card>
  );
}

export default ProjectCard;
  
```

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

Tables

SCHEMAS

- cs157a
- sakila
- Scope
- Tables
- Course
- CourseHasProjects
- Instructor
- InstructorTeachCourses
- MilestoneHasReviews
- Milestones
- Project
- ProjectHasMilestones
- Reviews
- Student
- StudentHasCourses
- StudentHasProjects
- StudentHasTeams
- Team
- TeamHasReviews
- Views
- Stored Procedures
- Functions

Object Info

Table: Project

Columns:

project_id	int(10) AI PK	
project_name	varchar(60)	
project_description	text	
1	Three Tier Architecture Application	In this project, you will apply concepts presented.
2	Online Diagnostic Lab Reporting System	Nullam sed metus molestie, laoreet ex ut, pellen...
3	Online tuition management system	Donec in mi et lorem luctus dictum nec at odio...
4	Voice and speech based browser	Nullam sed metus molestie, laoreet ex ut, pellen...
6	Inventory management for android	Donec in mi et lorem luctus dictum nec at odio...
7	Character recognition system	Donec in mi et lorem luctus dictum nec at odio...
8	Emergency Call System in Android	Donec in mi et lorem luctus dictum nec at odio...
9	Bluetooth based attendance system	Donec in mi et lorem luctus dictum nec at odio...
10	Automated DAL generator in Java	Donec in mi et lorem luctus dictum nec at odio...
11	Online appointment system	Integer sed velit fermentum metus sodales luctu...
12	Medical Order sheet for android	Contusion and laceration of left cerebrum with ...
13	Help Desk	Toxic effect of other corrosive organic compound...
14	Financial Calculator for android	Nulla a scelerisque orci. Aliquam erat volupat...
15	Barcode reader with android device	Nulla a scelerisque orci. Aliquam erat volupat...

Action Output

Time	Action	Response
2 14:38:45	SELECT * FROM Scope.StudentHasCourses LIMIT 0, 1000	225 row(s) returned
3 14:49:08	SELECT * FROM Scope.Project LIMIT 0, 1000	14 row(s) returned

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

The screenshot shows the MySQL Workbench interface. On the left, the schema browser displays the 'Scope' database with its tables, including 'CourseHasProjects'. The main area shows the results of a query:

```
1 • SELECT * FROM Scope.CourseHasProjects;
```

course_id	project_id
278917254	1
278917254	2
278917254	3
299655012	4
426383441	6
426383441	7
55116348	8
55116348	9
55116348	10
55116348	11
55116348	12
55116348	13
55116348	14

Below the table, the 'Action Output' section shows two queries:

Action	Time	Action	Response
3	14:49:08	SELECT * FROM Scope.Project LIMIT 0, 1000	14 row(s) returned
4	14:49:18	SELECT * FROM Scope.CourseHasProjects LIMIT 0, 1000	13 row(s) returned

Query Completed

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

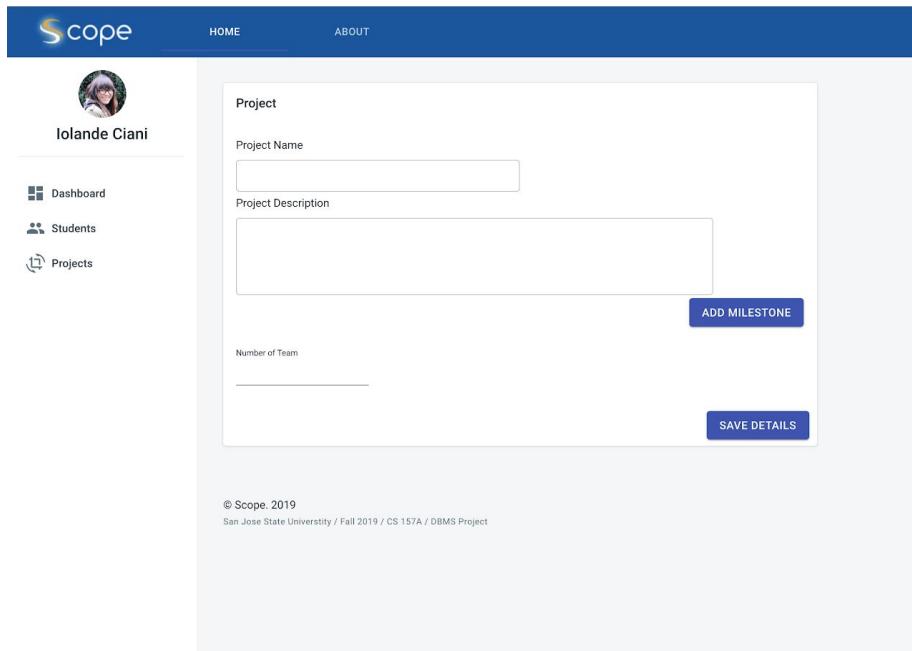
```

42
43
44
45 /**
46 * {POST} Get all project information from one course
47 * @param {string} course_id The id of the course
48 * @return {MySQL result} A collection of the project in the course; each contains the id of the project, project name and project descriptio
49 */
50 router.post('/course-project', function (req, res) {
51   var course_id = req.headers.course_id;
52   if (!course_id) {
53     res.status(401).send("Missing course_name")
54   }
55   var sql = "SELECT project_id , project_description, project_name FROM CourseHasProjects JOIN Course USING (course_id) JOIN Project US
56   connection.query(sql, course_id, function (err, result) {
57     if (err) throw err
58     res.send(result)
59   })
60 }

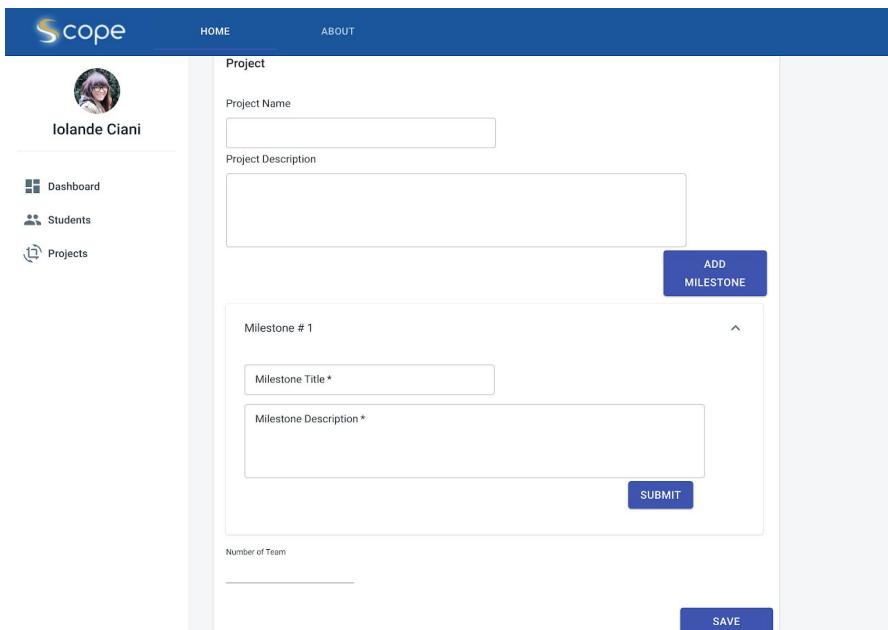
```

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

Add Project & Add Milestones Front-End

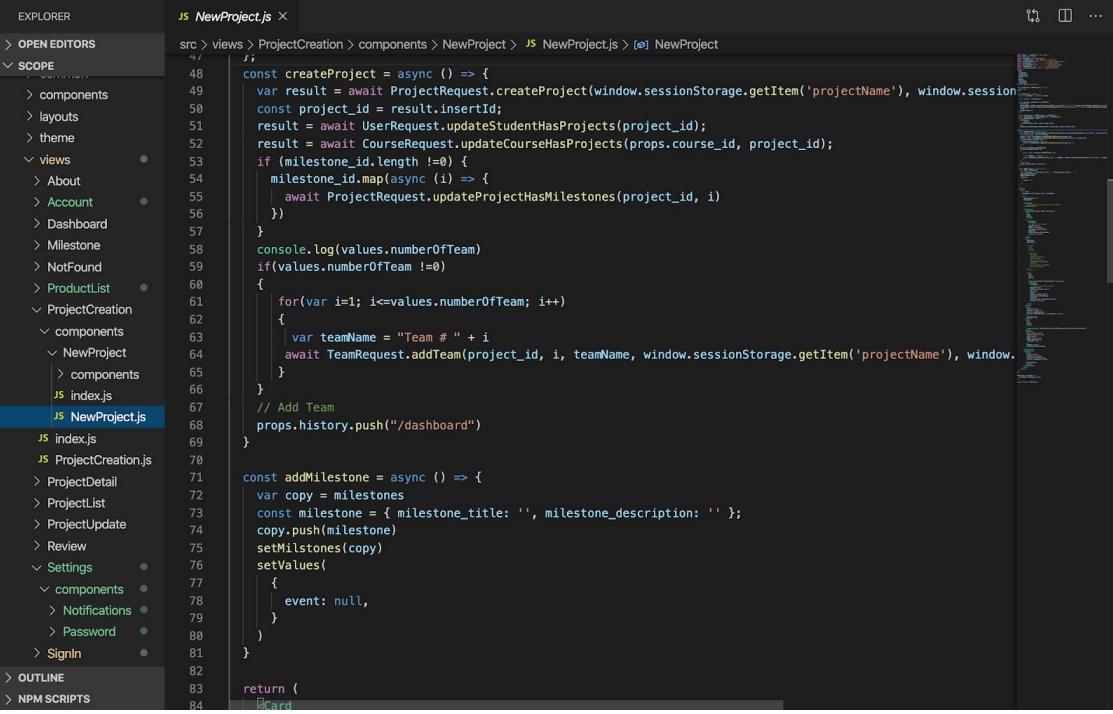


The above screenshot shows the pop up window that displays when the user clicks on the 'Add Project' button on the top right corner of the dashboard. This popup page contains a grid which includes text fields where the user can enter the project name, project description, add milestones, specify number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

Front-End Code



A screenshot of a code editor showing the `NewProject.js` file. The code is written in JavaScript and contains logic for creating a project and adding milestones. The code editor has a dark theme with syntax highlighting. The left sidebar shows a tree view of the project structure, including components like `About`, `Account`, `Dashboard`, `Milestone`, `NotFound`, `ProductList`, `ProjectCreation`, `ProjectDetail`, `ProjectList`, `ProjectUpdate`, `Review`, `Settings`, `Notifications`, `Password`, and `SignIn`. The `NewProject` component is selected in the tree view.

```
JS NewProject.js
src > views > ProjectCreation > components > NewProject > JS NewProject.js > NewProject
47
48 const createProject = async () => {
49     var result = await ProjectRequest.createProject(window.sessionStorage.getItem('projectName'), window.session
50
51     const project_id = result.insertID;
52     result = await UserRequest.updateStudentHasProjects(project_id);
53     result = await CourseRequest.updateCourseHasProjects(props.course_id, project_id);
54
55     if (milestone_id.length !=0) {
56         milestone_id.map(async (i) => {
57             await ProjectRequest.updateProjectHasMilestones(project_id, i)
58         })
59     }
60
61     console.log(values.numberOfTeam)
62     if(values.numberOfTeam !=0)
63     {
64         for(var i=1; i<=values.numberOfTeam; i++)
65         {
66             var teamName = "Team # " + i
67             await TeamRequest.addTeam(project_id, i, teamName, window.sessionStorage.getItem('projectName'), window.
68         }
69     }
69
70     // Add Team
71     props.history.push("/dashboard")
72
73
74
75
76
77
78
79
80
81
82
83
84
```

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

```

EXPLORER JS NewProject.js ...
OPEN EDITORS src > views > ProjectCreation > components > NewProject > JS NewProject.js > [e] NewProject
SCOPE
  > components 135
  > layouts 136
  > theme 137
  > views 138
    > About 139
    > Account 140
    > Dashboard 141
    > Milestone 142
    > NotFound 143
    > ProductList 144
  > ProjectCreation 145
  > components 146
    > NewProject 147
      > components 148
        > index.js 149
        > NewProject.js 150
        > index.js 151
        > ProjectCreation.js 152
        > index.js 153
        > NewProject.js 154
        > index.js 155
        > ProjectCreation.js 156
        > ProjectDetail 157
        > ProjectList 158
        > ProjectUpdate 159
        > Review 160
        > Settings 161
        > components 162
          > Notifications 163
          > Password 164
          > SignIn 165
        > index.js 166
        > NewProject.js 167
        > index.js 168
        > ProjectCreation.js 169
      > index.js 170
    > outline 171
  > NPM SCRIPTS

```

<Grid>

<Textfield>

<ExpansionPanel>

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

Tables

project_id	project_name	project_description
1	Three Tier Architecture Application	In this project, you will apply concepts presented...
2	Online Diagnostic Lab Report	Nullam sed metus molestie, laoreet ex ut, pellen...
3	Online tuition management system	Donec in mi et lorem luctus dictum nec at odio...
4	Voice and speech based browser	Nullam sed metus molestie, laoreet ex ut, pellen...
5	Inventory management for android	Donec in mi et lorem luctus dictum nec at odio...
6	Character recognition system	Donec in mi et lorem luctus dictum nec at odio...
7	Emergency Call System in Android	Donec in mi et lorem luctus dictum nec at odio...
8	Bluetooth based attendance system	Donec in mi et lorem luctus dictum nec at odio...
9	Automated DAL generator in Java	Donec in mi et lorem luctus dictum nec at odio...
10	Online appointment system	Integer sed velit fermentum metus sodales luctu...
11	Medical Order sheet for android	Contusion and laceration of left cerebrum with i...
12	Help Desk	Toxic effect of other corrosive organic compound...
13	Financial Calculator for android	Nulla a scelerisque orci. Aliquam erat volutpat...
14	Barcode reader with android	Nulla a scelerisque orci. Aliquam erat volutpat...
15		
NULL	NULL	NULL

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

The screenshot shows the MySQL Workbench interface. On the left, the Schemas tree view shows the 'Scope' schema expanded, revealing tables like Course, Instructor, Project, etc. The central area displays a query editor with the following SQL command:

```
1 • SELECT * FROM Scope.ProjectHasMilestones;
```

The result grid shows the data from the ProjectHasMilestones table:

project_id	milestone_number
1	3
2	4
2	5
2	6
2	7
3	8
3	9
4	10
4	11
4	12
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
	HULL
	HULL

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.

The screenshot shows the MySQL Workbench interface. On the left, the Schema browser displays the 'sakila' database structure, including tables like 'Course', 'CourseHasProjects', 'Instructor', etc., and the 'Scope' schema which contains the 'Milestones' table. The main area shows a query editor with the following SQL:

```
1 * | SELECT * FROM Scope.Milestones;
```

The results grid shows the data from the 'Milestones' table:

	milestone_number	milestone_title	milestone_description
1	Project Proposal	Propose a Three-tier Database Application and I...	
2	Project Design	Each team must submit an ER Model for the ap...	
3	Project Final Report	Identify key features of web-based application ar...	
4	Milestone_Title_04	Quisque in tincidunt augue. Sed ullamcorper tell...	
5	Milestone_Title_05	Vivamus sit amet arcu gravida, suscipit velit qui...	
6	Milestone_Title_06	Sed lobortis nibh, iaculis eget turpis vitae, Interdu...	
7	Milestone_Title_07	Proin dictum nibh id posuere convallis. Donec p...	
8	Milestone_Title_08	Mauris eget dolor imperdiet massa vehicula pha...	
9	Milestone_Title_09	Pellentesque rutrum justo metus, ac venenatis l...	
10	Milestone_Title_10	In hac habitasse platea dictumst. Etiam non pul...	
11	Milestone_Title_11	Integer quis nisi ac enim maximus fringilla vel id...	
12	Milestone_Title_12	Duis eget est vel turpis aliquam ullamcorper. Mo...	
13	Milestone_Title_13	Cras at nibh tristique felis pretium aliquet vitae e...	
14	Milestone_Title_14	Pellentesque fringilla, est et porta rutrum, ipsum...	
15	Milestone_Title_15	Sed sodales urna in libero pellentesque suscipit...	
45	ms	ms	
NULL	NULL	NULL	

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 screenshot shows a code editor with the file 'project.js' selected. The code is as follows:

```
> OPEN EDITORS
  server > routes > JS project.js > ...
  ↵
  ↵ 68  */
  ↵ 69  router.post('/add-project', function (req, res) {
  ↵ 70    var project_name = req.body.project_name;
  ↵ 71    var project_description = req.body.project_description;
  ↵ 72    if (!project_name || !project_description) {
  ↵ 73      return res.status(401).send("Missing Information")
  ↵ 74    }
  ↵ 75    var sql = "INSERT INTO Project (project_name, project_description) VALUES (?, ?)"
  ↵ 76    var projectTable = [project_name, project_description]
  ↵ 77    connection.query(sql, projectTable, function (err, result) {
  ↵ 78      if (err) {
  ↵ 79        print(err)
  ↵ 80      }
  ↵ 81      res.send(result)
  ↵ 82    })
  ↵ 83  })
  ↵ 84
```

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

```
JS team.js      85  /**
JS user.js      86   * {POST} Insert a new instance into ProjectHasMilestones table
> views          87   * @param {string} project_id The id of the project
JS app.js        88   * @param {string} milestone_number The id of milestone that a project contains
JS key.js        89   * @return {MySQL result} MySQL successful / unsuccessful insertion message
{} package-lock.json 90   */
{} package.json  91   */
> src           92 router.post('/updateProjectHasMilestones', function (req, res) {
  ● 93   var project_id = req.body.project_id;
  ● 94   var milestone_number = req.body.milestone_number;
  ● 95   if (!project_id || !milestone_number) {
  ● 96     return res.status(401).send("Missing Information")
  ● 97   }
  ● 98   var sql = "INSERT INTO ProjectHasMilestones (project_id, milestone_number) VALUES (?, ?)"
  ● 99   connection.query(sql, [project_id, milestone_number], function (err, result) {
  100     if (err) {
  101       print(err)
  102     }
  103     res.send(result)
  104   })
  105 }
```

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

Delete Project

Front-End

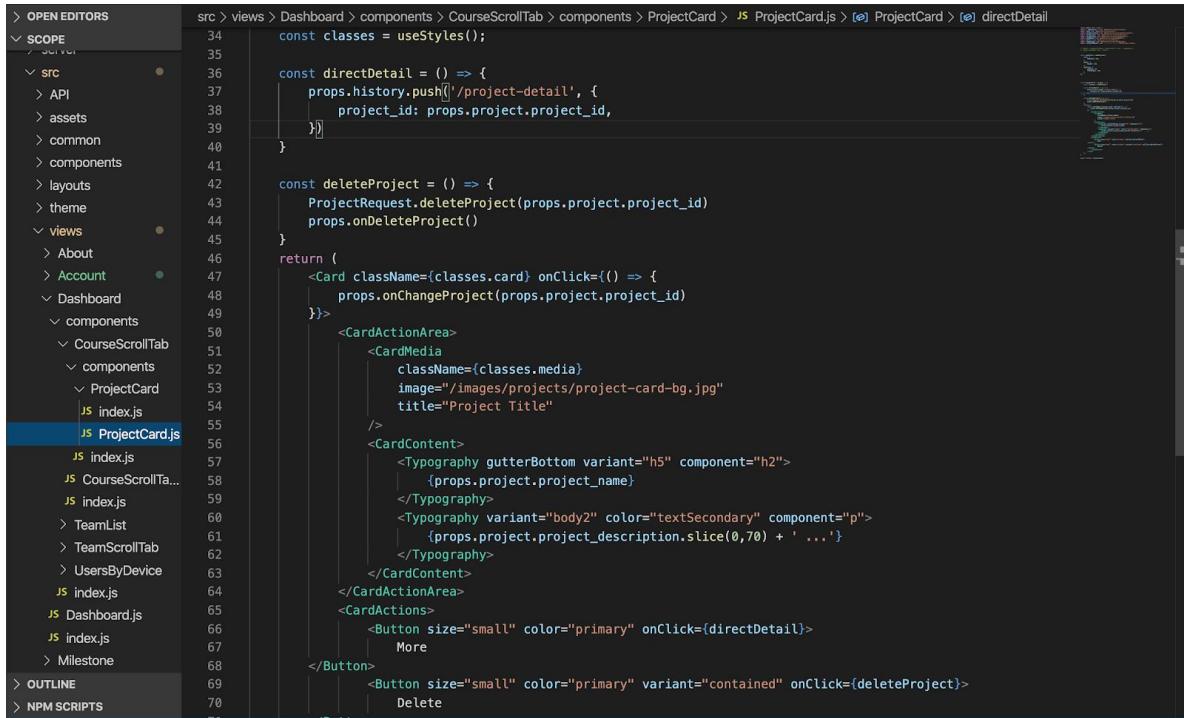
The screenshot shows the Scope application's front-end interface. At the top, there is a navigation bar with the logo "Scope", "HOME", "ABOUT", and a user icon. Below the navigation bar is a sidebar on the left with a profile picture of Iolande Ciani, her name, and links for "Dashboard", "Students", and "Projects". The main content area is titled "Courses" and shows a list of courses: CS157A, CS166, CS159, CS149, CMPE172, CS134, and CS152. Under each course, there is a thumbnail image of a blue and yellow geometric design, a project title, a brief description, and two buttons: "MORE" and "DELETE". The "CS166" course is currently selected. In the "Courses" section, there are two projects listed: "Voice and speech based browser" and "Sample Project". The "Sample Project" has a "DELETE" button below it.

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

This screenshot shows the same Scope application interface as the previous one, but with a key difference: the "Sample Project" has been deleted from the CS166 course. The "Courses" section now only lists the "Voice and speech based browser" project under CS166, with its "DELETE" button still visible.

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

Front-End Code



```
> OPEN EDITORS
  ✓ SCOPE
    ✓ src
      > API
      > assets
      > common
      > components
      > layouts
      > theme
    ✓ views
      > About
      > Account
    ✓ Dashboard
      ✓ components
        ✓ CourseScrollTab
          ✓ components
            ✓ ProjectCard
              JS index.js
              JS ProjectCard.js
              JS index.js
              JS CourseScrollTa...
              JS index.js
        > TeamList
        > TeamScrollTab
        > UsersByDevice
        JS index.js
        JS Dashboard.js
        JS index.js
        > Milestone
  > OUTLINE
  > NPM SCRIPTS

src > views > Dashboard > components > CourseScrollTab > components > ProjectCard > JS ProjectCard.js > [e] ProjectCard > [e] directDetail
  const classes = useStyles();
  const directDetail = () => {
    props.history.push('/project-detail', {
      project_id: props.project.project_id,
    })
  }
  const deleteProject = () => {
    ProjectRequest.deleteProject(props.project.project_id)
    props.onDeleteProject()
  }
  return (
    <Card className={classes.card} onClick={() => {
      props.onChangeProject(props.project.project_id)
    }}>
      <CardActionArea>
        <CardMedia
          className={classes.media}
          image="/images/projects/project-card-bg.jpg"
          title="Project Title"
        />
        <CardContent>
          <Typography gutterBottom variant="h5" component="h2">
            {props.project.project_name}
          </Typography>
          <Typography variant="body2" color="textSecondary" component="p">
            {props.project.project_description.slice(0,70) + ' ...'}
          </Typography>
        </CardContent>
      </CardActionArea>
      <CardActions>
        <Button size="small" color="primary" onClick={directDetail}>
          More
        </Button>
        <Button size="small" color="primary" variant="contained" onClick={deleteProject}>
          Delete
        </Button>
      </CardActions>
    </Card>
  )
}

export default ProjectCard
```

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

Tables

The screenshot shows the MySQL Workbench interface. On the left, the Schemas tree shows the Scope schema selected. In the main area, a query editor window displays the following SQL command:

```
1 • SELECT * FROM Scope.Project;
```

The result grid shows the following data:

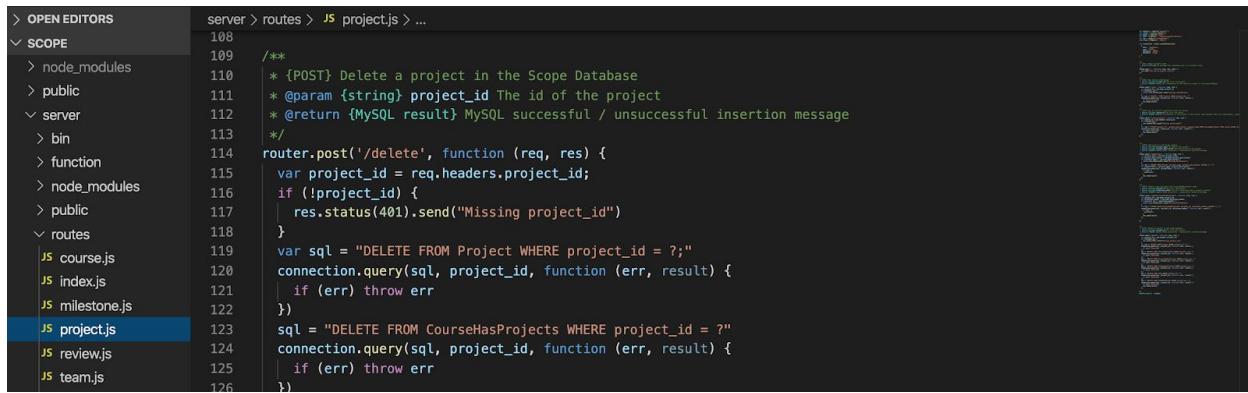
project_id	project_name	project_description
1	Three Tier Architecture Appli...	In this project, you will apply concepts presented...
2	Online Diagnostic Lab Report...	Nullam sed metus molestie, laoreet ex ut, pellen...
3	Online tuition management sy...	Donec in mi et lorem luctus dictum nec at odio...
4	Voice and speech based bro...	Nullam sed metus molestie, laoreet ex ut, pellen...
6	Inventory management for an...	Donec in mi et lorem luctus dictum nec at odio...
7	Character recognition system	Donec in mi et lorem luctus dictum nec at odio...
8	Emergency Call System In An...	Donec in mi et lorem luctus dictum nec at odio...
9	Bluetooth based attendance s...	Donec in mi et lorem luctus dictum nec at odio...
10	Automated DAL generator in...	Donec in mi et lorem luctus dictum nec at odio...
11	Online appointment system	Integer sed velit fermentum metus sodales luctu...
12	Medical Order sheet for android	Contusion and laceration of left cerebrum with l...
13	Help Desk	Toxic effect of other corrosive organic compound...
14	Financial Calculator for android	Nulla a scelerisque orci. Aliquam erat volutpat...
15	Barcode reader with android...	Nulla a scelerisque orci. Aliquam erat volutpat...
48	Sample Project	Sample project description
HULL	HULL	HULL

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

This screenshot is identical to the one above, showing the MySQL Workbench interface with the Scope schema selected. The query editor and result grid are the same, displaying the 16 rows of sample project data. The difference is that the row for the Sample Project (project_id 48) is no longer present in the result grid, indicating it has been deleted.

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

Backend Code



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar titled 'OPEN EDITORS' containing a tree view of project files. The file 'project.js' is currently selected and highlighted with a blue bar at the bottom of the sidebar. The main area displays the code for 'project.js'. The code is written in JavaScript and interacts with a MySQL database via a connection object. It includes comments explaining the purpose of each section, such as deleting a project by ID or updating course-project associations.

```
> OPEN EDITORS
  < SCOPE
    > node_modules
    > public
  < server
    > bin
    > function
    > node_modules
    > public
  < routes
    JS course.js
    JS index.js
    JS milestone.js
    JS project.js
    JS review.js
    JS team.js
  server > routes > JS project.js > ...
108  /**
109   * {POST} Delete a project in the Scope Database
110  * @param {string} project_id The id of the project
111  * @return {MySQL result} MySQL successful / unsuccessful insertion message
112  */
113  router.post('/delete', function (req, res) {
114    var project_id = req.headers.project_id;
115    if (!project_id) {
116      res.status(401).send("Missing project_id")
117    }
118    var sql = "DELETE FROM Project WHERE project_id = ?;"
```

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

Add Review Front-End

The screenshot shows the Scope application's interface for adding a review. At the top, there is a blue header bar with the 'Scope' logo, 'HOME', and 'ABOUT' links. On the left, a sidebar menu includes 'Dashboard', 'Students', and 'Projects'. The main content area is titled 'MY REVIEW' and shows a list of 'Reviewee - Iolande Ciani' and 'Reviewee - Hanni Arro'. For each reviewee, there is a table with milestones and dropdown menus for selecting performance levels. A dropdown menu for 'Milestone # 1' for Hanni Arro shows options: 'None', 'Unsatisfactory', 'Improvement needed', 'Meets expectations', and 'Exceeds expectations', with 'Meets expectations' being selected. Below the table, there is a text area labeled 'Enter Your Review Here...' containing the note: 'The meetings Timothy leads often run beyond their scheduled time. Over the next year, Timothy needs to make sure that his meetings begin and end on time.' At the bottom right of the form is a 'SAVE REVIEW' button.

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

```

    function callback()
    {
        props.callback()
        setExpanded(false);
    }

    return (
        <div className={classes.root}>
            {milestone != null ? milestone.map((milestone, index) => (
                const panelName = "panel" + index + 1
                return <ExpansionPanel expanded={expanded === panelName} key={index} onChange={handleChange(panelName)}>
                    <ExpansionPanelSummary
                        expandIcon=<ExpandMoreIcon />
                        aria-controls="panel1bh-content"
                        id="panel1bh-header"
                    >
                        <Typography className={classes.heading}>{`Milestone # ${getMilestoneTitle(index)}`}</Typography>
                        <Typography className={classes.secondaryHeading}>{getMilestoneTitle(index)}</Typography>
                        <Typography className={classes.secondaryHeading}>{getReview(index + 1) != null ? rating.getReview(index + 1).review_description : null}</Typography>
                    </ExpansionPanelSummary>
                    <ExpansionPanelDetails>
                        <ExpansionForm updateForm={callback} review_id={getReview(index + 1) != null ? getReview(index + 1).review_id : null}>
                            <Typography>
                                <CheckCircleOutline />
                            </Typography>
                            <Typography>
                                /* {getReview(index + 1) != null ? <Rating name="read-only" style={{ marginTop: 30, marginLeft: 60 }} /> : null} */
                            </Typography>
                        </ExpansionForm>
                    </ExpansionPanelDetails>
                </ExpansionPanel>
            )) : null}
        </div>
    );
}

```

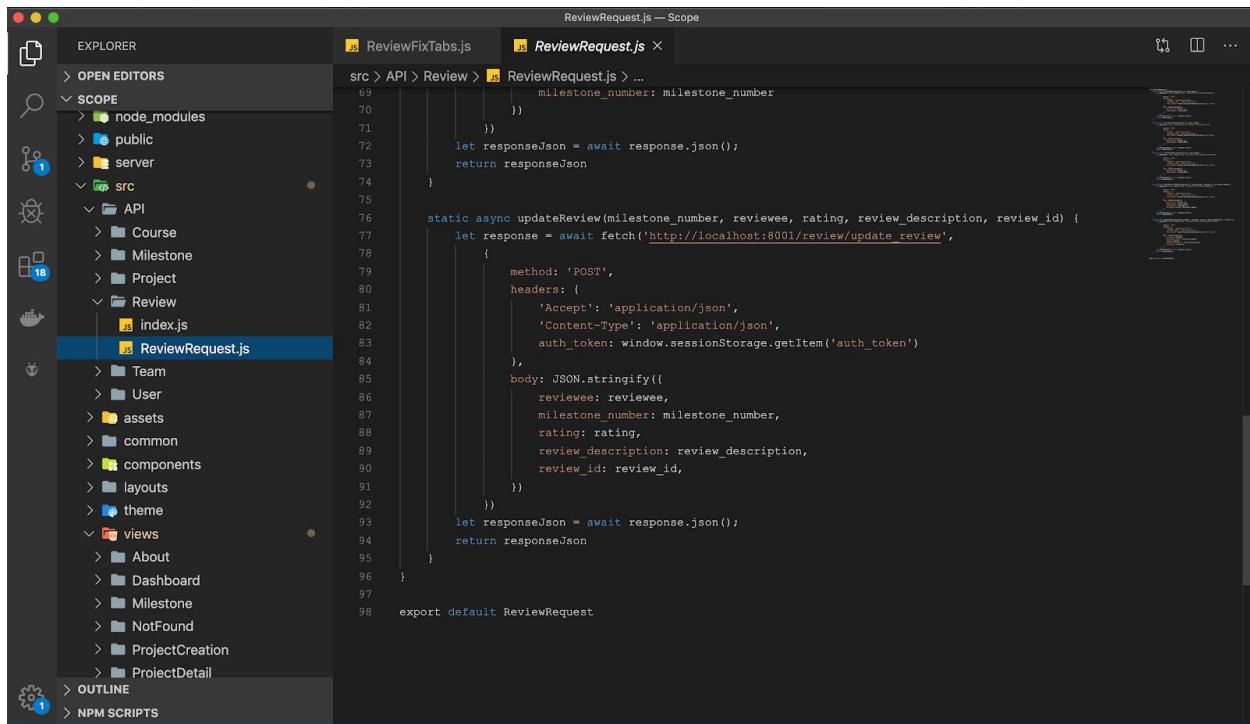
Tables

MySQL Workbench

Local Instance 3306

review_id	milestone_number	reviewer	reviewee	rating	review_description
1	1	106996496	169135057	1	The meetings Timothy leads often run beyond their scheduled time. Over the next year, Tim...
2	1	169135057	169135057	2	Larry is in need of working on his ability to cooperate with fellow co-workers. There have...
3	1	217021555	169135057	3	Jim continues to be a valued member of our crew and is a person we are able to count on...
4	2	106996496	169135057	1	Heather does not show up for work on time nor does she work a normal work schedule.
5	2	169135057	169135057	2	Tim is extremely dependable with regard to his attendance and regularly exhibits punctua...
6	2	217021555	169135057	3	Kevin has had several shouting episodes with his supervisor over the last review period...
7	3	106996496	169135057	0	None
8	3	169135057	169135057	2	Janet starts every day refreshed and ready for any problems she may face throughout the...
9	3	217021555	169135057	3	Matt is clearly a "people person" that always communicates to his clients how much he e...
10	1	217021555	106996496	1	Jim's team has vastly improved their overall performance over the last year due to his po...
11	1	106996496	106996496	3	James almost always goes above and beyond his normal job activities to please his cust...
12	1	169135057	106996496	4	A very important skill that Sally has is her ability to effectively communicate topics that ve...
13	2	217021555	106996496	1	Frank has an ability to effectively communicate change of plans to employees and keeps...
14	2	106996496	106996496	1	James almost always goes above and beyond his normal job activities to please his cust...
15	2	169135057	106996496	4	A very important skill that Sally has is her ability to effectively communicate topics that ve...
16	3	217021555	106996496	1	Hilda's project team always raves at her ability to encourage open communication among...
17	3	106996496	106996496	0	None
18	3	169135057	106996496	4	A very important skill that Sally has is her ability to effectively communicate topics that ve...
19	1	169135057	217021555	0	None
20	1	217021555	217021555	4	Has made frequent errors that are harmful to business operations.
21	1	106996496	217021555	4	Jim needs to work on his ability to receive feedback from his co-workers. Usually Jim ten...
22	2	169135057	217021555	4	George should try to open himself up with team members and stop isolating himself from...
23	2	217021555	217021555	4	Julie needs to work on improving her communication skill set when it comes to discussin...
24	2	106996496	217021555	4	Shirley has done an outstanding job this past year cooperating with her team members d...
25	3	169135057	217021555	4	Heather always displays a positive attitude when working with her group members.

Backend Code



The screenshot shows a code editor interface with the following details:

- EXPLORER:** On the left, the project structure is displayed under the "SCOPE" section. It includes "node_modules", "public", "server", and a "src" folder containing "API", "Course", "Milestone", "Project", "Review" (which contains "index.js" and "ReviewRequest.js"), "Team", "User", "assets", "common", "components", "layouts", "theme", "views" (containing "About", "Dashboard", "Milestone", "NotFound", "ProjectCreation", and "ProjectDetail"), and "OUTLINE" and "NPM SCRIPTS".
- TAB BAR:** The tabs show "ReviewFixTabs.js" and "ReviewRequest.js".
- EDITOR:** The "ReviewRequest.js" tab is active, displaying the following code:

```
ReviewRequest.js — Scope
src > API > Review > ReviewRequest.js > ...
69     milestone_number: milestone_number
70   })
71   let responseJson = await response.json();
72   return responseJson
73 }
74 }
75
76 static async updateReview(milestone_number, reviewee, rating, review_description, review_id) {
77   let response = await fetch('http://localhost:8001/review/update_review',
78   {
79     method: 'POST',
80     headers: {
81       'Accept': 'application/json',
82       'Content-Type': 'application/json',
83       auth_token: window.sessionStorage.getItem('auth_token')
84     },
85     body: JSON.stringify({
86       reviewee,
87       milestone_number,
88       rating,
89       review_description,
90       review_id,
91     })
92   })
93   let responseJson = await response.json();
94   return responseJson
95 }
96
97
98 export default ReviewRequest
```

View Milestones Front-End

The screenshot shows the Scope application's homepage. On the left is a sidebar with a user profile picture of Iolande Ciani, her name, and three navigation links: Dashboard, Students, and Projects. The main content area displays a project card for "Three Tier Architecture Application". The card includes a title, a short description, and a table of milestones:

Milestone #1	Project Proposal
Milestone #2	Project Design
Milestone #3	Project Final Report

At the bottom of the card, there are social sharing icons (heart, share) and a collapse/expand arrow. The footer contains the copyright information: © Scope 2019, San Jose State University / Fall 2019 / CS 157A / DBMS Project.

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.

The screenshot shows the Scope application's project detail page for "Three Tier Architecture Application". The layout is identical to the homepage, with the sidebar on the left and the detailed project card on the right. The card includes the project title, a detailed description, and the same table of milestones. The description in the card states: "Each team must submit an ER Model for the application. Based on your project functional requirements: ØIdentify the entities, attributes, dependences, relationships, constraints, etc. ØShow ISA, multi-way relationship weak entity sets, etc. that apply to your design. ØExplanation for each entity set and relationship, write a short description in plain English of what it represents or models. One or two sentences per entity set and relationship is enough. These descriptions are primarily to help understand what you are modeling." The footer contains the copyright information: © Scope 2019, San Jose State University / Fall 2019 / CS 157A / DBMS Project.

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

Tables

The screenshot shows the MySQL Workbench interface with the 'Schemas' tab selected. Under the 'Scope' schema, the 'Tables' section is expanded, and the 'Milestones' table is selected. A SQL query window at the top displays the command: 'SELECT * FROM Scope.Milestones;'. The results grid below shows 15 rows of data with columns: 'milestone_number', 'milestone_title', and 'milestone_description'. The data includes various milestones such as 'Project Proposal', 'Project Design', and 'Testing'.

	milestone_number	milestone_title	milestone_description
1	1	Project Proposal	Propose a Three-tier Database Application and i...
2	2	Project Design	Each team must submit an ER Model for the ap...
3	3	Project Final Report	Identify key features of web-based application ar...
4	4	Milestone Title_04	Quisque in lacinia augue. Sed ullamcorper telli...
5	5	Milestone Title_05	Vivamus sit amet arcu gravida, suscipit velit qui...
6	6	Milestone Title_06	Sed libero nibh, lacinia eget turpis vitae. Interdu...
7	7	Milestone Title_07	Proin dictum nunc id posuere convallis. Donec p...
8	8	Milestone Title_08	Mauris egestas dolor imperdiet massa vehicula pha...
9	9	Milestone Title_09	Pellentesque rutrum justo metus, ac viventibus ...
10	10	Milestone Title_10	In hac habitasse platea dictumst. Etiam non pul...
11	11	Milestone Title_11	Integer quis nisi ac enim maximus fringilla vel id...
12	12	Milestone Title_12	Duis egestas vel turpis aliquam ullamcorper. Mo...
13	13	Milestone Title_13	Cras at nibh tristique felis pretium aliquet vita e...
14	14	Milestone Title_14	Pellentesque fringilla, est et porta rutrum, ipsum...
15	15	Milestone Title_15	Sed sodales urna in libero pellentesque suscipit...
	45	Testing	Testing
	NULL	NULL	NULL

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

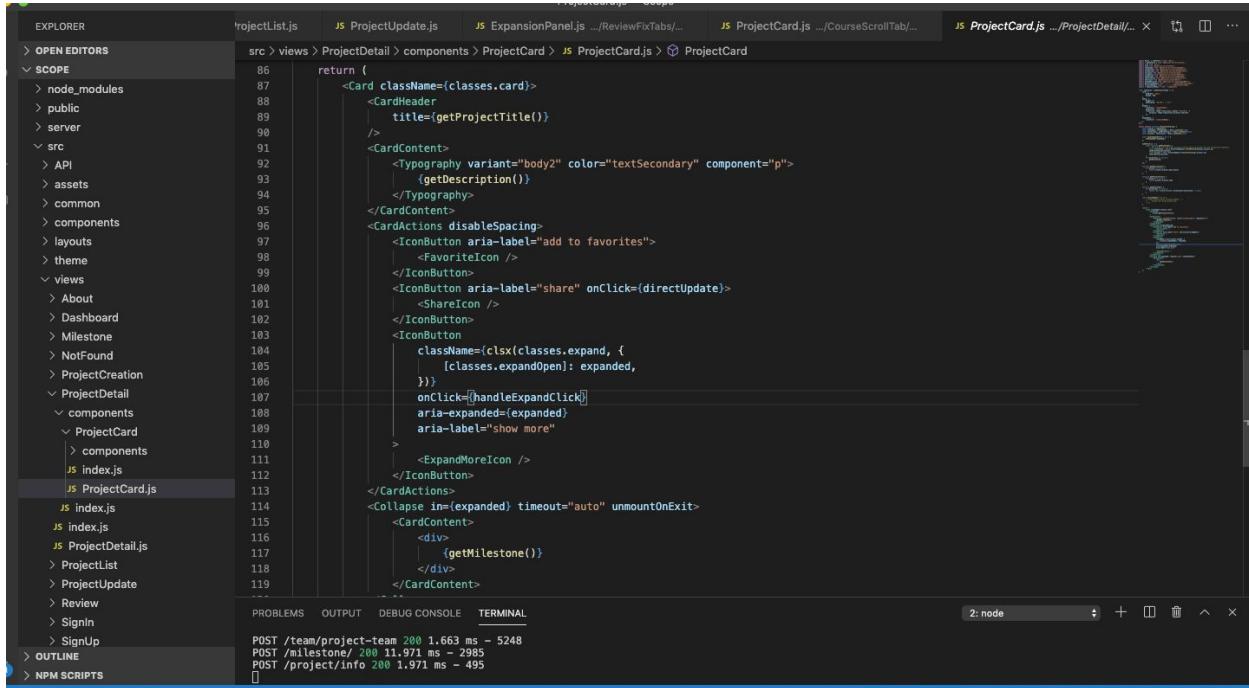
The screenshot shows the MySQL Workbench interface with the 'Schemas' tab selected. Under the 'Scope' schema, the 'Tables' section is expanded, and the 'ProjectHasMilestones' table is selected. A SQL query window at the top displays the command: 'SELECT * FROM Scope.ProjectHasMilestones;'. The results grid below shows 15 rows of data with columns: 'project_id' and 'milestone_number'. The data indicates specific milestones assigned to projects.

	project_id	milestone_number
1	1	1
1	1	2
1	1	3
2	2	4
2	2	5
2	2	6
2	2	7
3	3	8
3	3	9
4	4	10
4	4	11
4	4	12
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11
12	12	12
13	13	13
14	14	14
15	15	15
	NULL	NULL

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

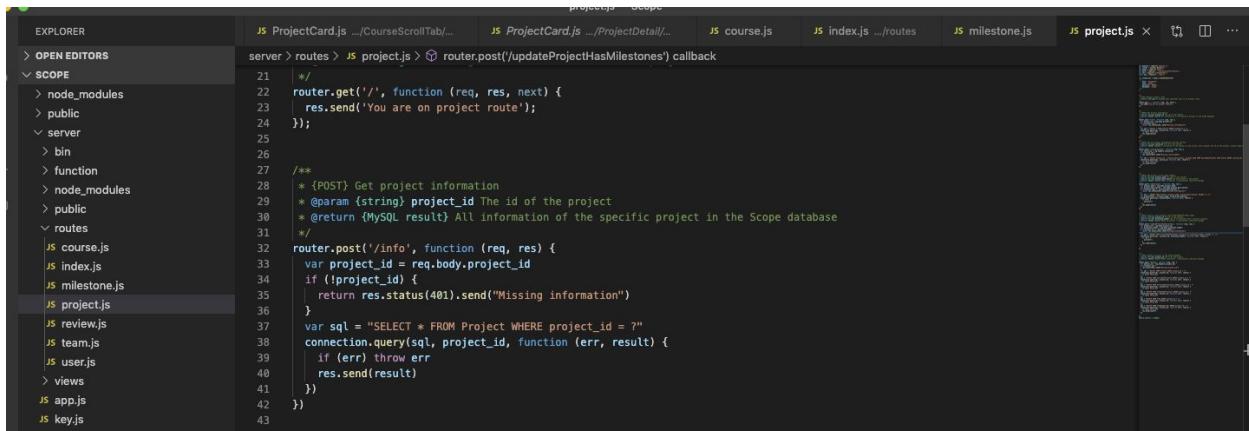


```

EXPLORER          ProjectList.js   JS ProjectUpdate.js   JS ExpansionPanel.js .../ReviewFixTabs/...   JS ProjectCard.js .../CourseScrollTab/...   JS ProjectCard.js .../ProjectDetail/...
SCOPE           src > views > ProjectDetail > components > ProjectCard > JS ProjectCard.js > ProjectCard
src > views > ProjectDetail > components > ProjectCard > JS ProjectCard.js > ProjectCard
86   return (
87     <Card className={classes.card}>
88       <CardHeader
89         title={getProjectTitle()}
90       />
91       <CardContent>
92         <Typography variant="body2" color="textSecondary" component="p">
93           | {getDescription()}
94         </Typography>
95       </CardContent>
96       <CardActions disableSpacing>
97         <IconButton aria-label="add to favorites">
98           | <FavoriteIcon />
99         </IconButton>
100        <IconButton aria-label="share" onClick={directUpdate}>
101          | <ShareIcon />
102        </IconButton>
103        <IconButton
104          className={clsx(classes.expand, {
105            [classes.expandOpen]: expanded,
106          })}
107          onClick={handleExpandClick}
108          aria-expanded={expanded}
109          aria-label="show more"
110        >
111          <ExpandMoreIcon />
112        </IconButton>
113      </CardActions>
114      <Collapse in={expanded} timeout="auto" unmountOnExit>
115        <CardContent>
116          <div>
117            | {getMilestone()}
118          </div>
119        </CardContent>
120      </Collapse>
121    </Card>
122  )
123
```

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

Backend Code



```

EXPLORER          JS ProjectCard.js .../CourseScrollTab/...   JS ProjectCard.js .../ProjectDetail/...   JS course.js   JS index.js .../routes   JS milestone.js   JS project.js
SCOPE           server > routes > JS project.js > router.post('/updateProjectHasMilestones') callback
21   /**
22    * @param {string} project_id The id of the project
23    * @return {MySQL result} All information of the specific project in the Scope database
24  });
25
26 /**
27 * {POST} Get project information
28 * @param {string} project_id The id of the project
29 * @return {MySQL result} All information of the specific project in the Scope database
30 */
31 router.post('/info', function (req, res) {
32   var project_id = req.body.project_id
33   if (!project_id) {
34     return res.status(401).send("Missing information")
35   }
36   var sql = "SELECT * FROM Project WHERE project_id = ?"
37   connection.query(sql, project_id, function (err, result) {
38     if (err) throw err
39     res.send(result)
40   })
41 }
42 }
```

```
milestone.js — Scope
server > routes > JS milestone.js > ...
14  * {GET} Default milestone route
15  * @return {String} An message that indicates user is on milestone route
16  */
17  router.get('/', function (req, res, next) {
18  |   res.send('You are on milestone route');
19  });
20
21 /**
22  * {Post} Get project and project's milestones information
23  * @param {string} project_id The id of the project
24  * @return {MySQL result} A collection of object; each contains the name of the project, project description, course name
25  * milestone number, milestone title and milestone description
26  */
27
28 router.post('/', function (req, res) {
29  const project_id = req.body.project_id
30  if (!project_id) {
31    res.status(401).send("Missing Project ID")
32  }
33  const sql = "SELECT project_name, project_description, course_name, milestone_number, milestone_title, milestone_description FROM Mile
34  connection.query(sql, project_id, function (err, result) {
35    if (err) throw err
36    res.send(result)
37  })
38})
39
```

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 screenshot shows a web application interface for 'Scope'. At the top, there's a blue header bar with the 'Scope' logo, 'HOME', 'ABOUT', and a notifications/icon section. On the left, a sidebar for 'Iolande Ciani' lists 'Dashboard', 'Students', and 'Projects'. The main content area displays 'Scope2' (Team #2) with a 'REVIEW' button. It shows three team members: 'Lionello Keymer' (with a placeholder profile picture), 'Sheelagh Roslen' (with a placeholder profile picture), and 'Iolande Ciani' (with her actual profile picture). Below the members is a section for milestones:

Milestone #1	Project Proposal
Milestone #2	Project Design
Milestone #3	Project Final Report

At the bottom of the main content area, there's a copyright notice: '© Scope, 2019 San Jose State University / Fall 2019 / CS 157A / DBMS Project'.

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

```

src > views > Milestone > components > ProjectCard > JS ProjectCard.js > ProjectCard
  101     props.history.push('/project-update', {
  102       project_id: props.project_id,
  103       team_number: props.team_number
  104     })
  105   }
  106   return (
  107     <Card className={classes.card}>
  108       <CardHeader>
  109         title={getTeamTitle()}
  110         subheader={"Team # " + props.team_number}
  111       </CardHeader>
  112       <Button variant="contained" style={{ marginLeft: 1000, marginTop: -80 }} onClick={() => {
  113         directReview()
  114       }}>REVIEW</Button>
  115       <CardContent>
  116         <Typography variant="body2" color="textSecondary" component="p">
  117           {getTeamDescription()}
  118         </Typography>
  119         {/** Team Member List */
  120           getTeamMember() != null ? getTeamMember().map((m) => {
  121             return <div style={{padding: 10}}><Avatar alt="Remy Sharp" src={profile_pic[Math.floor(Math.random() * profile_pic.length)]} />
  122               // <Typography>m.student_firstname + " " + m.student_lastname</Typography>
  123           }) : null}
  124         </CardContent>
  125         <CardActions disableSpacing>
  126           <IconButton aria-label="add to favorites">
  127             <FavoriteIcon />
  128           </IconButton>
  129           <IconButton aria-label="share">
  130             <ShareIcon onClick={directUpdate}/>
  131           </IconButton>
  
```

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

project_id	team_number	student_id
1	1	106996496
1	1	169135057
1	1	217021555
1	2	362190333
1	2	373426725
1	3	399274333
1	3	439889584
4	1	106996496
1	4	625030840
1	4	628395928
1	5	648686743
1	5	755316052
6	1	106996496
1	6	784193319
2	1	784193319
2	2	784193319
3	1	784193319
3	2	784193319
4	1	784193319
4	2	784193319
1	1	106996496
1	1	106996496
1	1	106996496
1	1	106996496

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

Backend Code

```
> bin          33  |  })
> function      34  |})
> node_modules 35  |
> public        36  /**
  > routes         37  * {POST} Get a team's information
  | JS course.js   38  * @param {string} project_id The id of the project
  | JS index.js    39  * @param {string} team_number The team's number in a project
  | JS milestone.js 40  * @return {MySQL result} All information about specific team
  | JS project.js   41  */
  | JS review.js    42  router.post('/info', function (req, res) {
  | JS team.js       43  var project_id = req.body.project_id;
  | JS user.js        44  var team_number = req.body.team_number;
  > views           45  if (!project_id || !team_number) {
  JS app.js          46  |   return res.status(401).send("Missing Project ID")
  JS key.js          47  }
  {} package-lock.json 48  var sql = "SELECT * FROM Team WHERE project_id = ? AND team_number = ?";
  {} package.json     49  connection.query(sql, [project_id, team_number], function (err, result) {
  50  |   if (err) throw err
  51  |   res.send(result)
  52  |})
  53  })
  54
```

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

Edit Team & Add Team

Front-End

The screenshot shows the Scope application interface. At the top, there is a blue header bar with the word "Scope" on the left, and "HOME" and "ABOUT" buttons on the right. Below the header is a sidebar on the left containing a user profile picture of a person with long hair, the name "Iolande Ciani", and three navigation links: "Dashboard", "Students", and "Projects". The main content area displays a team named "Scope2" (Team #2). It includes a "REVIEW" button, a team description (a placeholder text about elementum), and three member profiles: "Lionello Keymer", "Sheelagh Roslen", and "Sheelagh Roslen" again. Below the members are two circular icons: a heart and a link symbol. At the bottom of the main content area, there are two buttons: "Milestone #1" and "Project Proposal".

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

The screenshot shows the Scope application interface with the "Edit Team" form open. The header and sidebar are identical to the previous screenshot. The main content area now shows a form for editing a team. It has fields for "Team Name" (containing "Scope2") and "Team Description" (containing a placeholder text about elementum). Below these fields is a "UPDATE TEAM" button. At the bottom of the page, there is a copyright notice: "© Scope, 2019" and "San Jose State University / Fall 2019 / CS 157A / DBMS Project".

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

The screenshot shows the Scope application's project creation interface. At the top, there is a navigation bar with links for HOME and ABOUT. On the left, a sidebar for user 'Iolande Ciani' includes a profile picture, a dashboard link, student and project lists, and a projects section. The main area is titled 'Project' and contains fields for 'Project Name' (set to 'Sample Project'), 'Project Description' (set to 'Sample project description'), and 'Number of Team' (set to '4'). There are buttons for 'ADD MILESTONE' and 'SAVE DETAILS'.

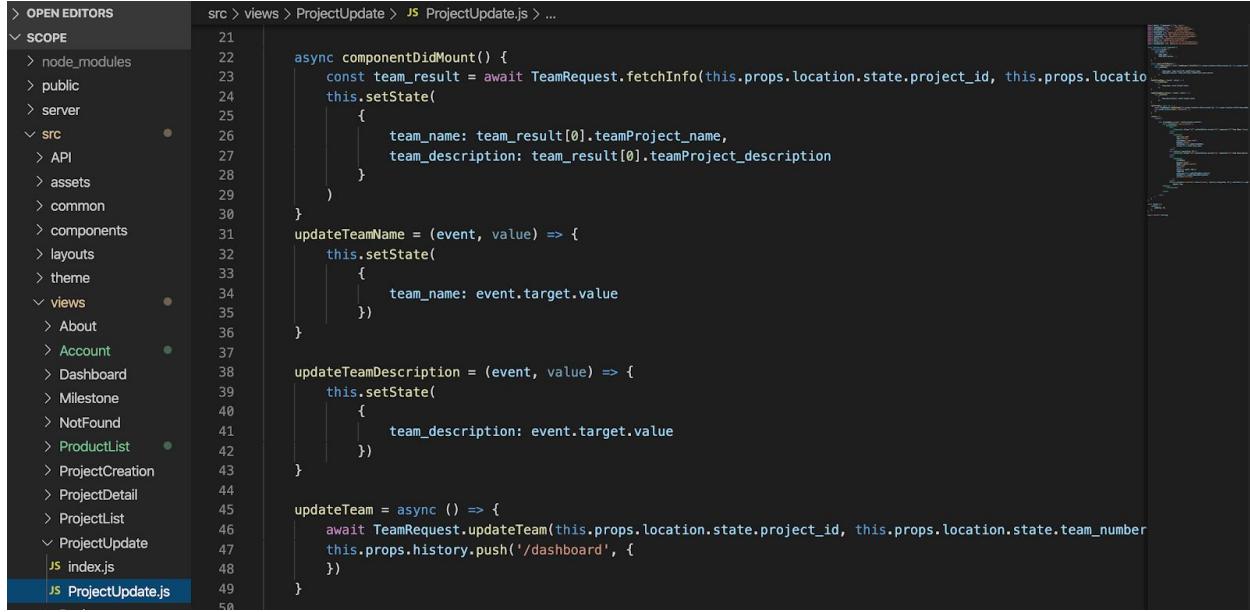
© Scope. 2019
San Jose State University / Fall 2019 / CS 157A / DBMS Project

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

The screenshot shows the Scope application's project list and team details. At the top, there is a navigation bar with links for HOME and ABOUT, and a search/filter icon. On the left, a sidebar for user 'Iolande Ciani' includes a profile picture, a dashboard link, student and project lists, and a projects section. The main area displays three projects: 'Inventory management for android', 'Character recognition system', and 'Sample Project'. Each project card has 'MORE' and 'DELETE' buttons. Below this, a 'Teams' section shows four teams: 'TEAM # 1', 'TEAM # 2', 'TEAM # 3', and 'TEAM # 4'. Each team has a preview image and a 'Sample Project' card with a 'Project Description' placeholder.

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

Front-end Code



```

> OPEN EDITORS          src > views > ProjectUpdate > JS ProjectUpdate.js > ...
< SCOPE                21
  > node_modules        22
  > public              23
  > server              24
  < src                  25
    > API                26
    > assets              27
    > common              28
    > components          29
    > layouts             30
    > theme               31
    < views               32
      > About              33
      > Account             34
      > Dashboard            35
      > Milestone            36
      > NotFound              37
      > ProductList           38
      > ProjectCreation       39
      > ProjectDetail          40
      > ProjectList            41
    < ProjectUpdate         42
      > index.js            43
      JS ProjectUpdate.js   44
      JS index.js           45
      JS ProjectUpdate.js   46
      JS index.js            47
      JS ProjectUpdate.js   48
      JS index.js           49
      JS ProjectUpdate.js   50

```

```

async componentDidMount() {
  const team_result = await TeamRequest.fetchInfo(this.props.location.state.project_id, this.props.location.state.team_number);
  this.setState({
    team_name: team_result[0].teamProject_name,
    team_description: team_result[0].teamProject_description
  })
}

updateTeamName = (event, value) => {
  this.setState({
    team_name: event.target.value
  })
}

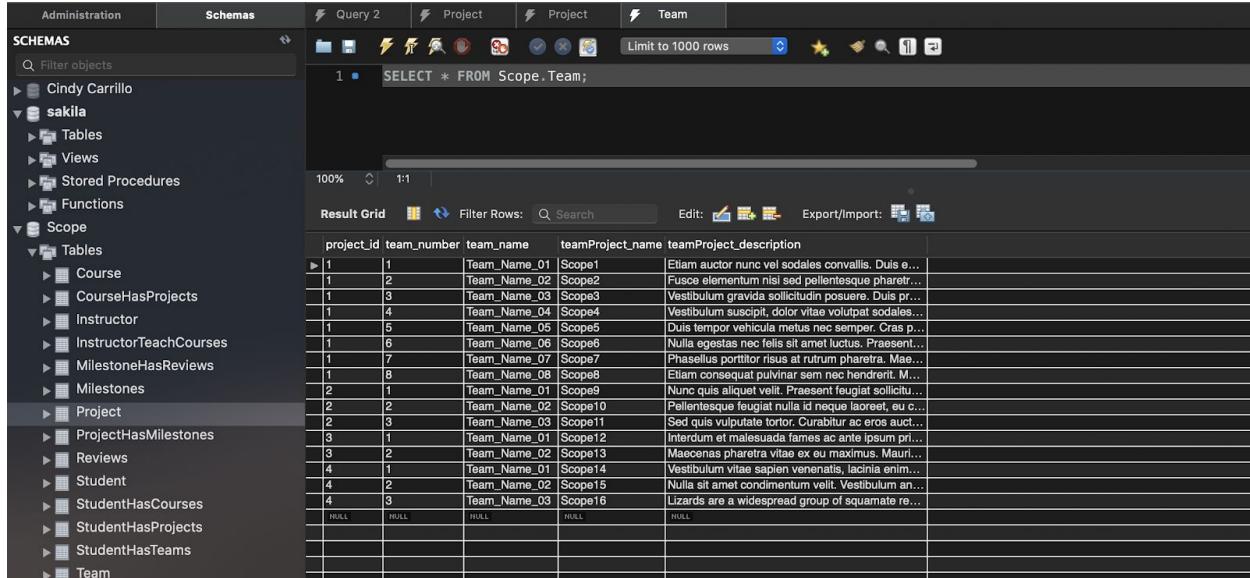
updateTeamDescription = (event, value) => {
  this.setState({
    team_description: event.target.value
  })
}

updateTeam = async () => {
  await TeamRequest.updateTeam(this.props.location.state.project_id, this.props.location.state.team_number);
  this.props.history.push('/dashboard', {
    });
}

```

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

Tables



project_id	team_number	team_name	teamProject_name	teamProject_description
1	1	Team_Name_01	Scope1	Etiam auctor nunc vel sodales convallis. Duis e...
1	2	Team_Name_02	Scope2	Fusce elementum nisi sed pellentesque pharetr...
1	3	Team_Name_03	Scope3	Vestibulum gravida sollicitudin posuere. Duis pr...
1	4	Team_Name_04	Scope4	Vestibulum suscipit, dolor vitae volutpat sodales...
1	5	Team_Name_05	Scope5	Duis tempor vehicula metus nec semper. Cras p...
1	6	Team_Name_06	Scope6	Nulla egestas nec felis sit amet luctus. Praesent...
1	7	Team_Name_07	Scope7	Phasellus porttitor risus at rutrum pharetra. Mae...
1	8	Team_Name_08	Scope8	Etiam consequat pulvinar sem nec hendrerit. M...
2	1	Team_Name_01	Scope9	Nunc quis aliquet velit. Praesent feugiat sollicu...
2	2	Team_Name_02	Scope10	Pellentesque feugiat nulla id neque laoreet, eu c...
2	3	Team_Name_03	Scope11	Sed quis vulputate tortor. Curabitur ac eros auct...
3	1	Team_Name_01	Scope12	Interdum et malesuada fames ac ante ipsum pri...
3	2	Team_Name_02	Scope13	Maecenas pharetra vitae ex eu maximus. Mauri...
4	1	Team_Name_01	Scope14	Vestibulum vitae sapien venenatis, lacinia enim...
4	2	Team_Name_02	Scope15	Nulla sit amet condimentum velit. Vestibulum an...
4	3	Team_Name_03	Scope16	Lizards are a widespread group of squamate re...
NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL

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.

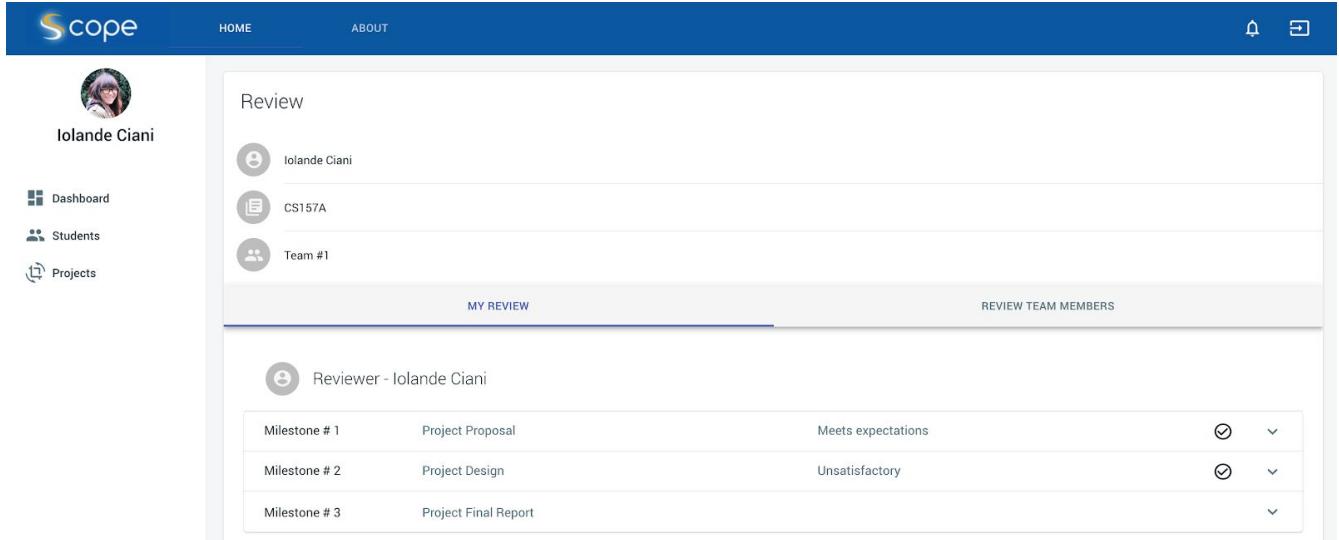
Backend Code

```
> bin                                58  * @param {string} team_number The team's number in a project
> function                             59  * @param {string} team_ProjectName The team's name
> node_modules                         60  * @param {string} team_ProjectDescription Team's own project description
> public                               61  * @return {MySQL result} MySQL successful / unsuccessful update message
> routes                               62  */
JS course.js                           63  router.post('/updateTeam', function (req, res) {
JS index.js                            64  var project_id = req.body.project_id;
JS milestone.js                         65  var team_number = req.body.team_number;
JS project.js                           66  var team_ProjectName = req.body.team_ProjectName;
JS review.js                            67  var team_ProjectDescription = req.body.team_ProjectDescription;
JS team.js                              68  if (!project_id || !team_number || !team_ProjectName || !team_ProjectDescription) {
JS user.js                             69  return res.status(401).send("Missing Project ID")
JS views                                70  }

var sql = "UPDATE Team SET teamProject_Name = ? ,teamProject_Description = ? WHERE project_id = ? AND team_numb
connection.query(sql, [team_ProjectName, team_ProjectDescription, project_id, team_number], function (err, res
| if (err) throw err
| res.send(result)
})                                         71
}                                            72
}                                              73
JS app.js                               74
JS key.js                               75
{} package-lock.json                   76
{} package.json                         77
> src                                  78
                                     79
```

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

A screenshot of the VS Code code editor showing the file "ReviewFixTabs.js" under the "REVIEW" component. The code is written in React and uses the TabPanel component from the "FullWidthTabs" library. The code handles the rendering of team members based on their index and direction.

```
ReviewFixTabs.js — Scope
src > views > Review > components > ReviewCard > components > ReviewFixTabs > ReviewFixTabs.js > FullWidthTabs > team_members.map((t) => {
  <TabPanel value=(value) index=(0) dir=(theme.direction)>
    <team_members != null ? team_members.map((t) => {
      return <div key=(t.student_id)>
        <Typography className=(classes.subTitle) variant="h4" component="h4">
          <ListItems>
            <ListItem>
              <ListItemAvatar>
                <Avatar>
                  <AccountCircle />
                </Avatar>
              </ListItemAvatar>
              <ListItemText primary="Course Name" />
              Reviewer - (t.student_firstname) (t.student_lastname)
            </ListItems>
          </Typography>
        </div>
    ) : null
  /* Item One */
</TabPanel>
<TabPanel value=(value) index=(1) dir=(theme.direction)>
  <team_members != null ? team_members.map((t) => {
    return <div key=(t.student_id)>
      <Typography className=(classes.subTitle) variant="h4" component="h4">
        <ListItems>
          <ListItem>
            <ListItemAvatar>
              <Avatar>
```

My Review Section

The screenshot shows the Scope application's 'My Review Section' for student Ioande Ciani. The left sidebar shows navigation options: Dashboard, Students, and Projects. The main area displays reviews from three reviewers:

- Reviewer - Ioande Ciani**
 - Milestone # 1: Project Proposal, Meets expectations. Comment: James almost always goes above and beyond his normal job activities to please his customers. James always demonstrates how excellent customer service has a lasting positive effect on customer relationships and further sales.
 - Milestone # 2: Project Design, Unsatisfactory.
 - Milestone # 3: Project Final Report, dropdown menu open.
- Reviewer - Hanni Arro**
 - Milestone # 1: Project Proposal, Exceeds expectations.
 - Milestone # 2: Project Design, Exceeds expectations.
 - Milestone # 3: Project Final Report, Exceeds expectations.
- Reviewer - Perry Lakes**
 - Milestone # 1: Project Proposal, Unsatisfactory.
 - Milestone # 2: Project Design, Unsatisfactory.
 - Milestone # 3: Project Final Report, Unsatisfactory.

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

Self Assessment

Reviewer - Current User

The screenshot shows a user interface for a project review system. At the top, there's a navigation bar with links for 'HOME' and 'ABOUT'. On the left, a sidebar displays the user's profile picture and name ('Iolande Ciani'), along with links for 'Dashboard', 'Students', and 'Projects'. The main content area is titled 'Reviewer - Iolande Ciani'. It lists three milestones with their descriptions and ratings:

Milestone #	Description	Rating
1	Project Proposal	Meets expectations
2	Project Design	Unsatisfactory
3	Project Final Report	(dropdown menu)

Below the milestones, a note states: "James almost always goes above and beyond his normal job activities to please his customers. James always demonstrates how excellent customer service has a lasting positive effect on customer relationships and further sales."

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

The screenshot shows a code editor with a dark theme. The left sidebar shows the project structure under 'SCOPE' (including 'NotFound', 'ProjectCreation', 'ProjectDetail', 'ProjectList', 'ProjectUpdate', 'Review', 'ReviewCard', 'ExpansionPanel', 'ReviewFixTabs', 'ExpansionForm', 'ExpansionPan...', and 'ExpansionPan...'). The right pane displays the code for 'ExpansionForm.js'.

```
ExpansionForm.js — Scope
Components > ReviewCard > components > ReviewFixTabs > components > ExpansionPanelForm > components > ExpansionForm > ExpansionForm.js
1  <div>
2   <div className={classes.selectList}>
3     <FormControl variant="outlined" className={classes.formControl}>
4       <InputLabel ref={(InputLabel) => demoSimpleSelectOutlinedLabel} id="demo-simple-select-outlined-label">
5         Select Overall Rating
6       </InputLabel>
7       <Select labelId="demo-simple-select-outlined-label" id="demo-simple-select-outlined" value={age} onChange={handleChange} labelWidth={labelWidth}>
8         <MenuItem value="">
9           <em>None</em>
10          </MenuItem>
11          <MenuItem value="1">Unsatisfactory</MenuItem>
12          <MenuItem value="2">Improvement needed</MenuItem>
13          <MenuItem value="3">Meets expectations</MenuItem>
14          <MenuItem value="4">Exceeds expectations</MenuItem>
15        </Select>
16        <FormHelperText>Required</FormHelperText>
17      </FormControl>
18    </div>
19    <TextField id="outlined-multiline-static" label="Enter Your Review Here" multiline rows="6" defaultValue="" className={classes.textField} margin="normal" variant="outlined" onChange={event => {
20      const { value } = event.target;
21      setInputValue(value)
22    }}>
23    </TextField>
24  </div>
```

Tables

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance 3306, Administration, Schemas.
- Tables:** cs157a, sakila, Scope (selected), Reviews, MilestoneHasReviews, TeamHasReviews.
- Scope Table Data:**

review_id	milestone_number	reviewer	reviewee	rating	review_description
1	1	106996496	169135057	1	The meetings Timothy leads often run beyond their scheduled time. Over the next year, T...
2	1	169135057	169135057	2	Larry is in need of working on his ability to cooperate with fellow co-workers. There have...
3	1	217021555	169135057	3	Jim continues to be a valued member of our crew and is a person we are able to count o...
4	2	106996496	169135057	1	Heather does not show up for work on time nor does she work a normal work schedule.
5	2	169135057	169135057	2	Tim is extremely dependable with regard to his attendance and regularly exhibits punctua...
6	2	217021555	169135057	3	Kevin has had several shouting episodes with his supervisor over the last review period....
7	3	106996496	169135057	0	Janet starts every day refreshed and ready for any problems she may face throughout th...
8	3	169135057	169135057	2	Matt is clearly a "people person" that always communicates to his clients how much he e...
9	3	217021555	169135057	3	Jim's team has vastly improved their overall performance over the last year due to his po...
10	1	217021555	106996496	1	James almost always goes above and beyond his normal job activities to please his cust...
11	1	106996496	106996496	3	Frank has an ability to effectively communicate change of plans to employees and keeps...
12	1	169135057	106996496	4	James almost always goes above and beyond his normal job activities to please his cust...
13	2	217021555	106996496	1	Heather does not show up for work on time nor does she work a normal work schedule.
14	2	106996496	106996496	1	Tim is extremely dependable with regard to his attendance and regularly exhibits punctua...
15	2	169135057	106996496	4	A very important skill that Sally has is her ability to effectively communicate topics that ve...
16	3	217021555	106996496	1	Hilda's project team always raves at her ability to encourage open communication among...
17	3	106996496	106996496	0	Julie needs to work on his ability to receive feedback from his co-workers. Usually Jim ten...
18	3	169135057	106996496	4	Shirley has done an outstanding job this past year cooperating with her team members d...
19	1	169135057	217021555	0	Heather always displays a positive attitude when working with her group members.
20	1	217021555	217021555	4	George should try to open himself up with team members and stop isolating himself from...
21	1	106996496	217021555	4	Julie needs to work on improving her communication skill set when it comes to discussin...
22	2	169135057	217021555	4	Shirley has done an outstanding job this past year cooperating with her team members d...
23	2	217021555	217021555	4	Heather always displays a positive attitude when working with her group members.
24	2	106996496	217021555	4	George should try to open himself up with team members and stop isolating himself from...
25	3	169135057	217021555	4	Julie needs to work on improving her communication skill set when it comes to discussin...

Backend Code

The screenshot shows a code editor with the following details:

- File:** ReviewRequest.js — Scope
- Content:**

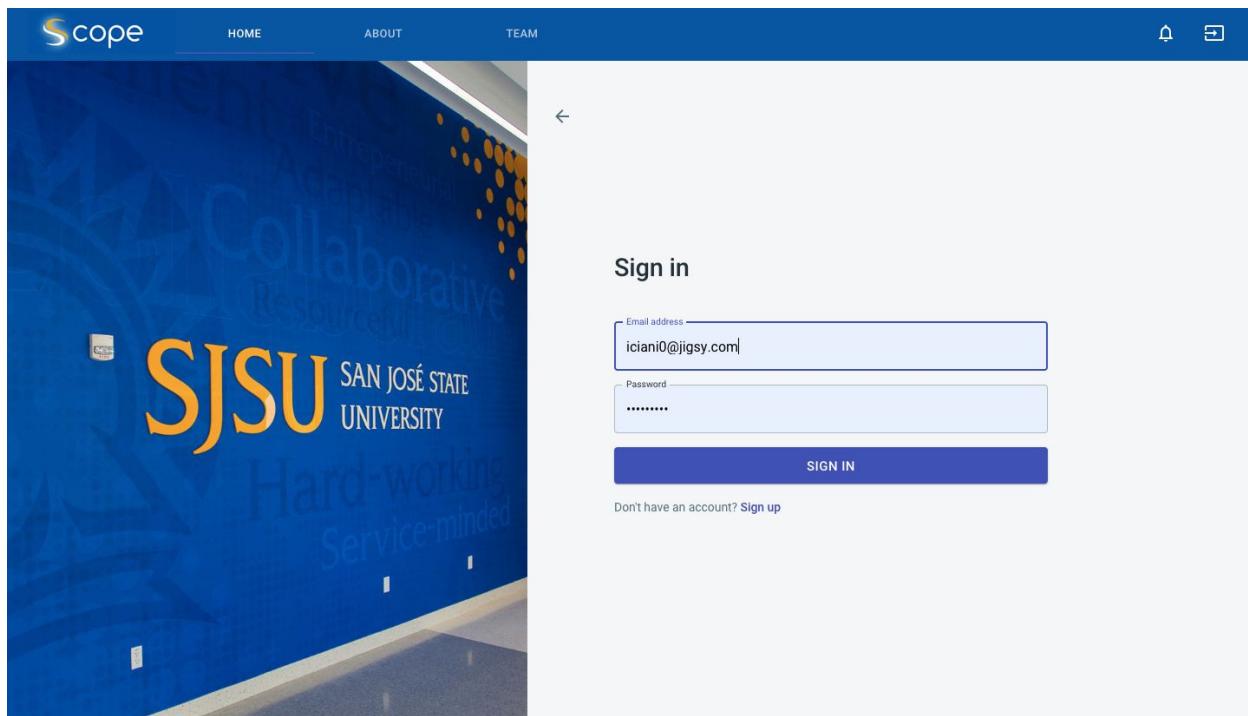
```

40   }
41   method: 'POST',
42   headers: {
43     'Accept': 'application/json',
44     'Content-Type': 'application/json',
45     auth_token: window.sessionStorage.getItem('auth_token')
46   },
47   body: JSON.stringify({
48     project_id: project_id,
49     team_number: team_number,
50   })
51 }
52 let responseJson = await response.json();
53 return responseJson
54 }

static async fetchSpecificReview(project_id, team_number, reviewer_id, milestone_number) {
  let response = await fetch('http://localhost:8001/review/specific',
  {
    method: 'POST',
    headers: {
      'Accept': 'application/json',
      'Content-Type': 'application/json',
      auth_token: window.sessionStorage.getItem('auth_token')
    },
    body: JSON.stringify({
      project_id: project_id,
      team_number: team_number,
      reviewer_id: reviewer_id,
      milestone_number: milestone_number
    })
  })
  let responseJson = await response.json();
  return responseJson
}

```

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 screenshot shows a code editor interface with several tabs open. The active tab is 'SignIn.js' under the 'src > views > SignIn' directory. The code is a functional component for handling sign-in logic. It includes a large block of JSX for rendering a button labeled 'Sign in' and a link labeled 'Sign up'. Below this is a form containing a password input field and a submit button. The code also includes prop types for 'history'.

```

    <Button
      className={classes.signInButton}
      color="primary"
      disabled={!formState.isValid}
      fullWidth
      size="large"
      type="submit"
      variant="contained"
      onClick={handleSignIn}
    >
      Sign in
    </Button>
    <Link
      component={RouterLink}
      to={{
        pathname: "/sign-up",
      }}
      variant="h6"
    >
      Sign up
    </Link>
    <Typography
      color="textSecondary"
      variant="body1"
    >
      Don't have an account?{' '}
    </Typography>
    <Form
      onSubmit={handleSignIn}
    >
      <Grid
        container
        spacing={2}
      >
        <Grid item>
          <Input
            type="password"
            value={formState.password}
            onChange={handlePasswordChange}
            required
          />
        </Grid item>
        <Grid item>
          <Button
            type="submit"
            variant="contained"
            onClick={handleSignIn}
          >Sign in</Button>
        </Grid item>
      </Grid>
    </Form>
  </div>
</div>
</Grid>
</Grid>
</div>
);

```

```

  SignIn.propTypes = {
    history: PropTypes.object
  };

```

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

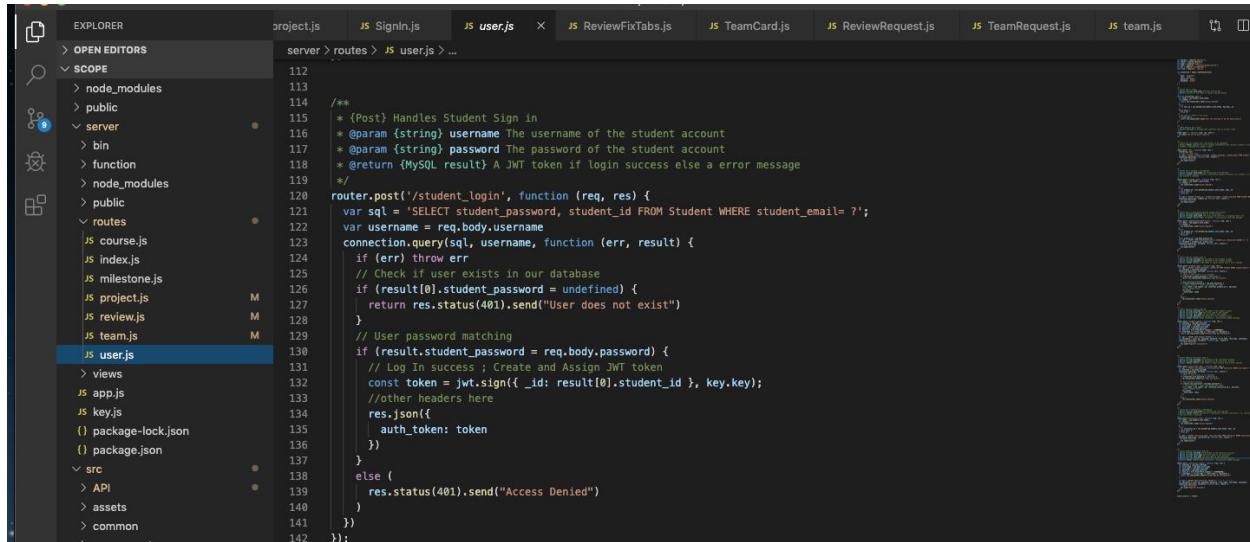
Table

The screenshot shows a database management interface with a sidebar for 'SCHEMAS' and a main area for a 'Result Grid'. The 'SCHEMAS' sidebar shows databases like 'cs157a' and 'sakila', and a 'Scope' schema containing tables such as 'Course', 'Instructor', 'Milestone', 'Project', 'Reviews', and 'Student'. The 'Result Grid' displays data from the 'Student' table. The columns are: student_id, student_username, student_password, student_firstname, student_lastname, and student_email. The data includes rows for various students with their respective details.

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@jgsy.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
217021558	plakes1	jsR\$4mt	Perry	Lakes	plakes1@paypal.com
362190333	lkeymer3	SwMK9LePEIq	Lionello	Keymer	lkeymer3@webnode.com
373426725	sroslen8	Ah45vuhq	Sheelagh	Roslen	sroslen8@mashable.com
399274333	pcandelin2	RAT5vV5jh8y	Putnem	Candelin	pcandelin2@domainmarket.com
439869584	gilles5	kNSHNKXTZ	Gabi	Ilie	gilles5@pcworld.com
625030840	bbitheila	DscfYN	Berti	Bitell	bbitheila@yellowbook.com
628395926	hsowtec	TjuwuJQnf	Horatius	Sowte	hsowtec@csmonitor.com
648866743	jpieritone	4CMno4	James	Pierton	jpieritone@csmonitor.com
755316052	okneeboned	bfxKUrw6bHiq	Consalve	Kneebone	okneeboned@kickstarter.com
784193319	mkohneke9	F4dxtr	Madelle	Kohneke	mkohneke9@pagesperso-orange.fr

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



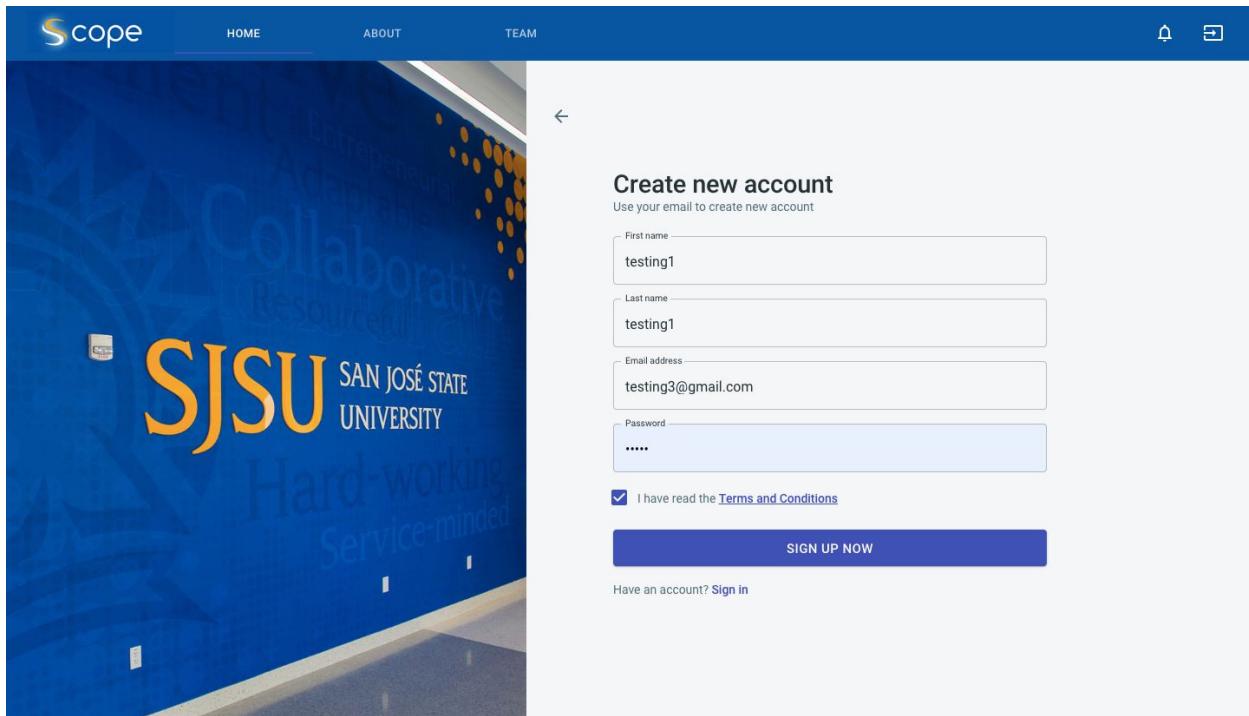
```
project.js JS Signin.js JS user.js X JS ReviewFixTabs.js JS TeamCard.js JS ReviewRequest.js JS TeamRequest.js JS team.js
server > routes > JS user.js > ...
112
113
114 /**
115 * {Post} Handles Student Sign in
116 * @param {string} username The username of the student account
117 * @param {string} password The password of the student account
118 * @return {MySQL result} A JWT token if login success else a error message
119 */
120 router.post('/student_login', function (req, res) {
121   var sql = 'SELECT student_password, student_id FROM Student WHERE student_email= ?';
122   var username = req.body.username
123   connection.query(sql, username, function (err, result) {
124     if (err) throw err
125     // Check if user exists in our database
126     if (result[0].student_password = undefined) {
127       return res.status(401).send("User does not exist")
128     }
129     // User password matching
130     if (result.student_password = req.body.password) {
131       // Log in success ; Create and Assign JWT token
132       const token = jwt.sign({_id: result[0].student_id}, key.key);
133       // /other headers here
134       res.json({
135         auth_token: token
136       })
137     }
138     else {
139       res.status(401).send("Access Denied")
140     }
141   });
142 });

123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
```

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

```

import React, { useState } from 'react';
import { IconButton, Typography, TextField } from '@material-ui/core';
import { classes } from 'classes';
import { constants } from 'constants';

const SignUp = () => {
  const [firstName, setFirstName] = useState('');
  const [lastName, setLastName] = useState('');

  const handleSignUp = () => {
    // Handle sign-up logic here
  };

  const handleFirstNameChange = (e) => {
    setFirstName(e.target.value);
  };

  const handleLastNameChange = (e) => {
    setLastName(e.target.value);
  };

  return (
    <div>
      <div>
        <IconButton onClick={handleBack}>
          <ArrowBackIcon />
        </IconButton>
      </div>
      <div>
        <form>
          <div>
            <Typography variant="h2">Create new account</Typography>
            <Typography color="textSecondary" gutterBottom>Use your email to create new account</Typography>
            <TextField
              className={classes.textField}
              error={hasError('firstName')}
              fullWidth
              helperText={
                hasError('firstName') ? formState.errors.firstName[0] : null
              }
              label="First name"
              name="firstName"
              onChange={handleFirstNameChange}
              type="text"
              value={formState.values.firstName || ''}
              variant="outlined"
            />
            <TextField
              className={classes.textField}
              error={hasError('lastName')}
              fullWidth
              helperText={
                hasError('lastName') ? formState.errors.lastName[0] : null
              }
              label="Last name"
              name="lastName"
              onChange={handleLastNameChange}
              type="text"
              value={formState.values.lastName || ''}
              variant="outlined"
            />
          </div>
          <div>
            <IconButton onClick={handleSignUp}>
              <ArrowForwardIcon />
            </IconButton>
          </div>
        </form>
      </div>
    </div>
  );
};

export default SignUp;
  
```

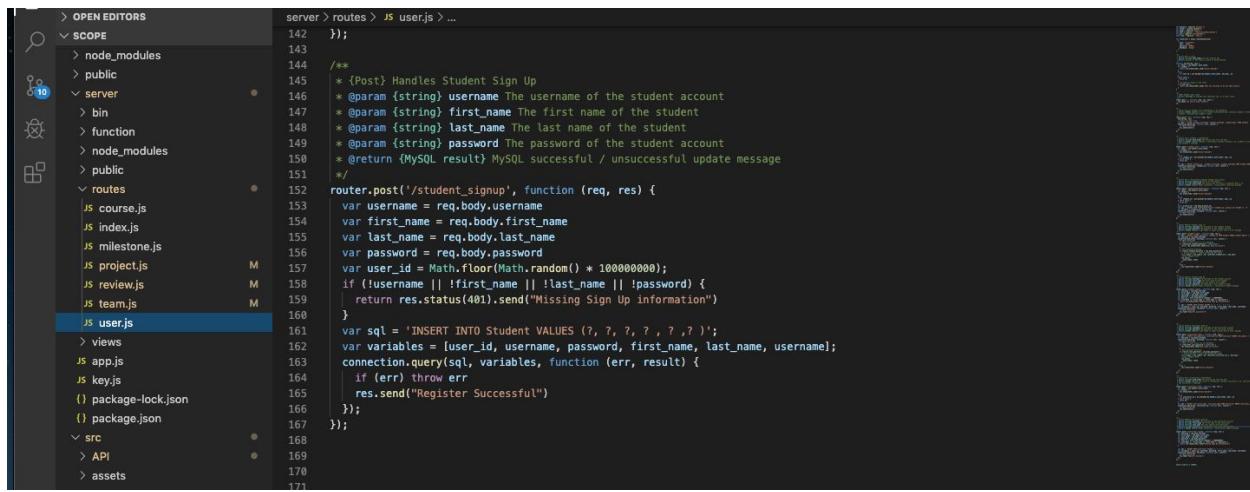
The above screenshot shows the corresponding font-end sign up page source code.

Table

student_id	student_username	student_password	student_firstname	student_lastname	student_email
102927927	rswinerd6	SmymQuvd	Rees	Swinerd	rswinerd6@unblog.fr
106996496	cianio0	A8bYvWJbx	Iolande	Ciani	cianio0@jigsy.com
123231231	testing	test	ing	sadad	adasd@gmail.com
16430249	testing3@gmail.com	cs157a	testing1	testing1	testing3@gmail.com
169135057	harro7	YX5ylTT5v	Hanni	Arro	harro7@posterous.com
217021555	piakes1	sRS4mt	Perry	Lakes	piakes1@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	bbitthella	DscfYN	Berti	Bitthell	bbitthella@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



```
server > routes > JS user.js > ...
142  });
143
144 /**
145  * {Post} Handles Student Sign Up
146  * @param {string} username The username of the student account
147  * @param {string} first_name The first name of the student
148  * @param {string} last_name The last name of the student
149  * @param {string} password The password of the student account
150  * @return {MySQL result} MySQL successful / unsuccessful update message
151 */
152 router.post('/student_signup', function (req, res) {
153  var username = req.body.username
154  var first_name = req.body.first_name
155  var last_name = req.body.last_name
156  var password = req.body.password
157  var user_id = Math.floor(Math.random() * 10000000);
158  if (!username || !first_name || !last_name || !password) {
159    return res.status(401).send("Missing Sign Up information")
160  }
161  var sql = 'INSERT INTO Student VALUES (?, ?, ?, ?, ?, ?)';
162  var variables = [user_id, username, password, first_name, last_name, username];
163  connection.query(sql, variables, function (err, result) {
164    if (err) throw err
165    res.send("Register Successful")
166  });
167 });
168
169
170
171
```

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 instructor_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 \rightarrow BC

A \rightarrow 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 \rightarrow BCDEF

A \rightarrow 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 useability. 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.