

# **Project Name: Automating Database Question Generation and Marking with PrairieLearn**

## **Project Charter, Scope & Requirements**

### **Version:**

Draft 1 - May 28th, 2023

### **Contacts:**

Matthew Obirek (matthewobirek@gmail.com),  
Skyler Alderson (skyler@thealdersons.org),  
Andrei Zipis (Andrei\_Zipis@hotmail.com),  
Nishant Srinivasan (nishant.srinivasan236@gmail.com)

# Table of Contents

1. Identification
2. Project Purpose
3. Project Objectives and Success Criteria
4. High Level Requirements
5. Assumptions
6. High Level Project Description
7. High Level Risks
8. Stakeholder List
9. Major Milestones
10. Budget
11. Client, Sponsor, Project Manager, Developers
12. Problem Statement
13. Requirements
  - 13.1. Functional Requirements
  - 13.2. Non-Functional Requirements
  - 13.3. Technical Requirements
  - 13.4. User Requirements
  - 13.5. Stretch Goals
14. Project Deliverables
15. Methodology / Workflow
16. UML Diagrams
17. Work Breakdown Structure
18. Approvals

# 1. Identification

The project - “Automating Database Question Generation and Marking with PrairieLearn”, is to successfully automate relational algebra and SQL question delivery and evaluation on PrairieLearn for students at UBC-O for our client, Dr. Lawrence. The project sponsor is Dr. Ramon Lawrence. The project manager is Nishant Srinivasan.

# 2. Project Purpose

As of right now, students in COSC 304 are reliant on using multiple different platforms for their coursework - including time-sensitive evaluations such as midterms and final examinations. These tools include Canvas, PrairieLearn, RelaX, GitHub, and a DBMS software to run and verify their queries locally.

The client would like to centralize all these tools in PrairieLearn for the students’ convenience and relieve them of untimely technical issues that may arise with using any or all of these tools at the same time.

# 3. Project Objectives and Success Criteria

Our objective is to integrate questions from specific, existing labs for COSC 304 from Canvas and GitHub into PrairieLearn, integrate the necessary tools for these labs such as RelaX, and automate randomized question generation as well as their evaluation process. In addition, the project requires us to implement a feature that allows students to verify their answers on PrairieLearn so that they need not rely on the DBMS software.

The team will add Python question generation and evaluation scripts for the specified labs and make front-end changes to allow usage of these scripts. The team will modify the user-interface by integrating RelaX into PrairieLearn and also implement a button that lets students verify their solutions without submitting.

# 4. High Level Requirements

- Students enrolled in COSC 304 at UBC-O will be able to complete labs 1, 2, and 3 entirely through PrairieLearn.
- Instructors teaching COSC 304 at UBC-O will be able to automatically generate problem questions from the first three labs in real time.
- All problems pertaining to the first three COSC 304 labs will be automatically marked once submitted.
- Submitting to PrairieLearn will be scalable with a large COSC 304 class size so that the service is not interrupted and performance does not degrade below acceptable levels.

- The user interface of the PrairieLearn questions will mimic the interface required to complete the first three labs.
  - RelaX will be ported for Lab 1
  - A textbox for DDL/SQL will be provided for Labs 2 & 3
    - UI will mimic SquirrelSQL

## 5. Assumptions

- Team is able to code in Python
- Team is able to code in JavaScript / Node.js
- Team is able to code in HTML and CSS
- Team is able to do front-end/back-end web development
- Team is able to work with Databases
- Team is able to use Docker
- Team maintains consistent contact with the client and provides them with weekly updates
- Timely and weekly scheduled meetings with client
- Team clearly communicates with the client about any issues and/or questions
- Questions for specified labs are integrated into PrairieLearn successfully
- RelaX is integrated into the PrairieLearn UI for appropriate labs
- Instructors are able to successfully automatically generate problems from specified labs
- Students are able to verify accuracy of queries before submitting
- Team meets deadlines

## 6. High Level Project Description

### Deliverable Description

- The deliverable will be an instance of PrairieLearn – software that allows for complex questioning and answering over the internet – hosted on local servers at UBC Okanagan Campus. The PrairieLearn instance will be able to ask and automatically answer Questions from COSC 304 lab 1 - Relational Algebra, COSC 304 lab 2 - SQL and MySQL table creation, and COSC 304 lab 3 – SQL and MySQL table querying.

### Elements to be Implemented

- For lab 1 - Relational Algebra, The RelaX user interface must be implemented, and an automatic marking of the string used.
- For lab 2 - SQL and MySQL table creation, a live instance of SQL must be query-able before the answer is sent for automatic marking
- For lab 3 - SQL and MySQL table querying, a live instance of SQL must be query-able before the answer is sent for automatic marking
- For all three labs, questions should be generalized - abstracted - for easy of automated generation
- For all three labs, as well as the previous PrairieLearn projects, local hosting at UBC Okanagan must be achieved

## 7. High Level Risks

- One or more team member(s) become unavailable
  - Sickness
  - Personal reasons
  - Bereavement
- One or more team member(s) refuse to work with the group or on project
- One or more team member(s) exaggerated capabilities
- Miscommunication between client and team
- Early errors not discovered until late in the project
- External events increase stress for all stakeholders
  - A pandemic occurs
  - A natural disaster occurs
- Project is delivered with unknown issues resulting in an unusable product
- PrairieLearn or RelaX are removed from GitHub prior to project completion

## 8. Stakeholder List

- Developers - Our Team
  - Project Manager: Nishant Srinivasan
  - Client Liaison: Matthew Obirek
  - Technical Lead: Skyler Alderson
  - Integration Lead: Andrei Zipis
- Clients
  - UBC Okanagan Campus
  - Dr Ramon Lawrence
- Maintainers
  - Dr.Ramon Lawrence
  - Future contributors/developers
- Users
  - University Professors
    - COSC 304,
    - Potentially COSC 111, 121
  - University Students
    - COSC 304,
    - Potentially COSC 111, 121

## 9. Major Milestones

1. May 29th, Week 3: First draft of charter, scope, and requirements documents
2. June 4th, Week 4: Final - Charter, scope, requirements, and preliminary design documents
3. June 9th, Week 4: Final - Design Documents
4. July 2nd, Week 8: Minimum viable product presentation
5. August 13th, Week 14: Completion of code, final report

## 10. Budget

This project has a monetary budget of \$0.

This project has a time budget of 64 to 80 people hours per week for fourteen weeks. It has an approximate total time budget of 900 hours for a team of four developers.

## 11. Owner, Project Manager, Developers

- Owner: Ramon Lawrence
- Project Manager: Nishant Srinivasan
- Client Liaison: Matthew Obirek
- Technical Lead: Skyler Alderson
- Integration Lead: Andrei Zipis

## 12. Problem Statement

The web service PrairieLearn does not currently support the ability for users to test SQL or relational algebra queries prior to submission. The automatic generation and marking of questions for SQL and relational algebra is not available to users on PrairieLearn.

## 13. Requirements

### 13.1. Functional Requirements

1. Provide documentation on how to deploy PrairieLearn on docker.
2. System will allow for relational algebra statements to be entered.
3. System will show visualizations of the resulting entered statement prior to submission.
4. System will automatically mark the relational algebra questions once submitted.
5. System will allow for DDL/SQL code to be entered.
6. System will show resulting tables of queries prior to submission.
7. System will automatically mark the DDL/SQL questions once submitted.

### 13.2. Non-Functional Requirements

1. PrairieLearn will be deployed with Docker.
2. The system will support 200 simultaneous user connections and submissions.
3. The system will ensure data integrity and preservation so that no data is lost upon submission.
4. The system will display entered queries within 1 second.
5. The system will return automarked submissions within 2 seconds.

6. The user interface will match existing software used for COSC 304.
7. The software will be maintainable. Code will be modular and appropriately commented.

### 13.3. Technical Requirements

1. Rebuild RelaX editor and calculator into PrairieLearn
2. Frontend: JavaScript, HTML, CSS
3. Backend: Python, Node.JS
4. Write JavaScript code that takes in SQL/DDI statements and displays appropriate table results
5. Write Python code that automatically marks submitted data and returns the students grade
6. Version control will be established with Github
7. CI/CD pipelines will be established using GitHub Actions & DroneCI to automate testing and deployment

### 13.4. User Requirements

#### 13.4.1. Student's User Requirements

1. Users will register to PrairieLearn System via CWL login.
2. Users will login to PrairieLearn System via CWL login.
3. Users will be able to view questions.
4. Users will receive visual feedback on entered statements prior to submission.
5. Users will be able to submit their answers.
6. Users will be able to view submitted answers and feedback.
7. Users will receive grades for submitted answers.

#### 13.4.2. Professor's User Requirements

1. Users will login to PrairieLearn System.
2. Users will select question types.
3. Users will use existing questions or automatically generate new ones.
4. Users will be able to see students' submissions.
5. Users will be able to see and edit students' grades.
6. Users will be able to set time limits.
7. Users will be able to set the number of allowed submissions.

### 13.5. Stretch Goals

1. System automatically generates relational algebra questions.
2. System automatically generates DDL/SQL questions.

## 14. Project Deliverables

1. Charter, Scope, Requirements, Planning documents
  - a. Use cases
  - b. Workflow
  - c. Work breakdown structure

2. RelaX integration, question generation, and auto marking
3. DDL integration, question generation, and auto marking
4. SQL integration, question generation, and auto marking

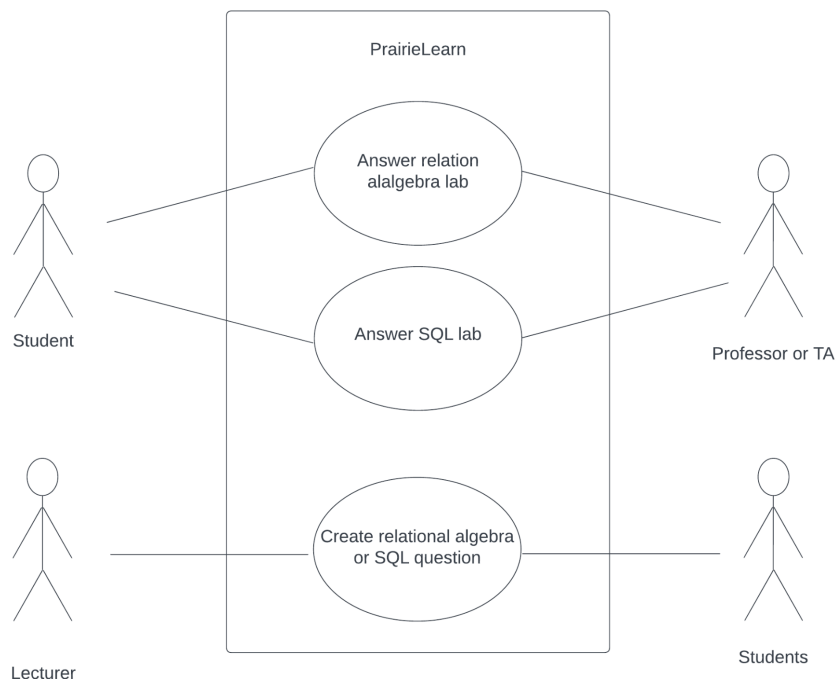
## 15. Methodology / Workflow

Our project will follow an agile methodology with a test-driven development (TDD) approach. All tests will be written before implementing each feature, ensuring that they first fail. We will be utilizing the Kanban framework through a Kanban board on GitHub projects to manage and visualize our workflow. Time tracking of tasks will be done using Clockify. The integrated development environment (IDE) we have chosen to use for both our front-end and back-end is VSCode. This is due to our familiarity with it and the many extensions it has allowing for a variety of programming languages and integration with GitHub and Docker. We will be using DroneCI for continuous integration (CI).

The group will be divided into two groups of two individuals per feature. This is due to the different requirements of integrating RelaX for Lab 1 and SQL/DDL for Labs 2 & 3. This division aims to optimize resource allocation while ensuring scrutiny and attention to detail for each feature.

## 16. UML (Use Case) Diagrams

### 16.1. Use Cases Diagram





**Figure 16.1:** A student may interact with the system (PrairieLearn) either through the Answer Relation Algebra Lab or Answer SQL Lab. The Professor or TA are able to see the student's grades after they have submitted their answer. Additionally, a Lecturer may create a new relational algebra or SQL question, which the students may answer.

#### 16.2. Use Case 1: Student Answers Relation Algebra Lab

Primary actor: Student

Description: A student wishes to complete their lab on relation algebra

Precondition: Student has successfully logged in to PrairieLearn and selected the relation algebra question.

Post condition: If the student successfully accessed the lab and submitted the question, then all questions will have been marked and the student would have received feedback.

Main Scenario:

1. Student selects the question
2. Student enters their answer in the RelaX editor
3. Students tests their query without submission
4. The output of the query is displayed for the student
5. The student submits their answer
6. The question is automatically graded and the student receives appropriate feedback

Extensions:

- 3a. The student's answer includes one or more formatting errors
  - 3a1. System issues an error message, informing the lecturer of the error and prevents submission
- 4a. Student is unsatisfied by the results of their test query
  - 4a1. Student returns to step 2, entering a new answer
- 5a. Student's answer includes one or more formatting errors
  - 5a1. System issues an error message, informing the lecturer of the error and prevents submission

#### 16.3. Use Case 2: Student Answers SQL Lab

Use case 2: Answer an SQL lab

Primary actor: Student

Description: A student wishes to complete their lab on SQL

Precondition: Student has successfully logged in to PrairieLearn and selected the relation algebra question.

Post condition: If the student successfully accessed the lab and submitted the question, then all questions will have been marked and the student would have received feedback.

Main Scenario:

1. Student selects the question

2. Student enters their answer in the SQL editor
3. Students tests their query without submission
4. The output of the query is displayed for the student
5. The student submits their answer
6. The question is automatically graded and the student receives appropriate feedback

Extensions:

- 3a. The student's answer includes one or more formatting errors
  - 3a1. System issues an error message, informing the lecturer of the error and prevents submission
- 4a. Student is unsatisfied by the results of their test query
  - 4a1. Student returns to step 2, entering a new answer
- 5a. Student's answer includes one or more formatting errors
  - 5a1. System issues an error message, informing the lecturer of the error and prevents submission

#### 16.4. Use Case 3: Lecturer Adds New Question

Use case 3: Creating a new relation algebra or SQL question

Primary actors: Lecturer

Description: During a lecture, the lecturer wishes to quiz the class on a relation algebra or SQL topic, so they create a new question

Precondition: The lecturer and students are all able to access PrairieLearn

Postcondition: The students have received automatic feedback on their responses and the lecturer may see the class' performance

Main scenario:

1. Lecturer selects question type as either relation algebra or SQL
2. Lecturer enters the question
3. Lecturer submits the question
4. Lecturer observes the students' performance using PrairieLearn

Extensions:

- 3a. Lecturer's question contains one or more errors
  - 3a1. System issues an error message, informing the lecturer of the error and prevents submission

## 17. Work Breakdown Structure

Tasks	Nishant	Andrei	Skyler	Matthew	Average Hours
<b>Assignments</b>					
Scope, Charter, & WBS	5	5	5	4	4.75

Design Documents	8	4	10	1	5.75
MVP Review and Prep.	6	6	5	4	5.25
Final Review and Prep.	6	6	7	4	5.75
<b>Meetings</b>					
Team	50	30	30	30	35
Client	10	10	10	10	10
Reporting	7	7	7	7	7
<b>Logs</b>					
Team	0	0	5	0	1.25
Dashboards	2	2	2	2	2
Individual	5	5	5	5	5
Communications	0	0	0	5	1.25
<b>Environment Setup</b>					
Finding Linter	1	1	3	1	1.5
Installing and Setting Up IDE's	1.5	1	1	2	1.375
Docker	15	15	15	30	18.75
<b>Familiarization</b>					
AutoER Documentation	4	5	3	5	4.25
Docker Deployment + Setup	10	10	10	10	10
Investigate Relax code	0	0	10	10	5
<b>PrairieLearn Setup</b>					
PL Documentation	3	5	3	5	4
Deploying PL to Docker	3	3	2	3	2.75
<b>Repo Management</b>					
Pull Requests	2	10	2	2.5	4.125

Branch Handling	1	1	0.5	2	1.125
Code Reviews	3	10	2	5	5
<b>Documentation</b>					
Project Board	5	0.5	0.5		2
Updating Charter and Scope Concurrently	1	1	1	0.5	0.875
PL Docker Deployment Documentation	0.5	0.5	0.5	5	1.625
Instructor Guide Documentation	0.5	0.5	0.5	1	0.625
<b>RelaX - Lab 1</b>					
Allow for Relational Algebra Statements to be Entered	0	0	20	30	12.5
Visualizations of the Resulting Entered Statement (Relation Algebra) prior to submission.	0	0	20	30	12.5
Automatically Mark the Relational Algebra Questions once Submitted.	0	0	10	10	5
Automatically Generate RelaX Questions.	0	0	10	0	2.5
<b>SQL/DDL</b>					
Allow for DDL/SQL Code to be Entered.	20	20	0	0	10
Visualizations of Resulting tables prior to submission.	3	2	0	0	1.25
Automatically Mark the DDL/SQL Questions Upon Submission.	10	10	0	0	5

Automatically Generate DDL/SQL Questions.	10	10	0	0	5
<b>Testing</b>					
Mock Data Generation	4	4	4	2	3.5
UI Tests	10	10	5	5	7.5
Unit Tests	18	20	20	10	17
Usability Tests	2	2	8	2	3.5
Integration Tests	3	3	3	3	3
	<b>229.5</b>	<b>219.5</b>	<b>240</b>	<b>246</b>	<b>234.25</b>

## 18. Approvals

---

Project Sponsor (UBC Okanagan)

---

Project Manager