Assignment 1: Charter, Scope and Requirements Document

Project 3: Building an Autograding Question System using PrairieLearn

University of British Columbia Okanagan

COSC 499 - Summer 2017

Date: May 27, 2022

Contacts:

Emiel van der Poel (emielvdpoel@gmail.com)

Luis Lucio (llucio99@icloud.com)

Prajeet Didden (prajeetdidden@gmail.com)

Siqiao Yuan (siqiaoyuan@gmail.com)

1. Project Objective

2. Stakeholder List

3. Major Milestones

4. Requirements

4.1 Functional Requirements

4.2 Non-Functional Requirements

4.3 Technical Requirements

4.4 User Requirements

4.4.1 Students User Requirements

4.4.2 Instructors User Requirements

5. Assumptions

6. High Level Risks

7. Methodology/Workflow

8. UML Diagrams

9. Work Breakdown Structure

10. Approvals

1. Project Objective

The objective of the project is to amend PrairieLearn, which is an open source software, to support a variety of questions regarding UML, SQL, and programming. The implementation will continue to use the previous front-end (AutoEr) which will be modified to use Mermaid JS for cleaner UML diagrams. Existing python question auto generation script will be imported into PrairieLearn which will replace the current existing backend (AutoEd). Investigations will be done for a possible integration of PrairieLearn into Canvas.

The current state of the system works with AutoEr functioning as the frontend, which lets students answer questions and see their grades. The backend works with AutoEd, which generates randomised questions from a python file. Documentation exists for the current system, and will need to be updated to incorporate the changes with the new system. The new backend system (PrairieLearn) will be deployed in a dockerized format, which will allow the professor full control over the system.

2. Stakeholder List

1. The University of British Columbia (Client Company).
2. Ramon Lawrence (Client).
3. Project Team.
4. Instructors (User).
5. Students (User).

3. Major Milestones

1. Finish documentation (August 13).
2. Design document (June 5).
3. Deliver design presentation (June 10).
4. Create early tests for test-driven design (June 18).
5. Update Nomnoml to Mermaid JS for rendering UML diagrams (June 25).
6. Import AutoEd back-end to PrairieLearn (July 1).
7. Create a working prototype / minimum viable product (July 8).
8. Import previous UML questions and design new SQL/programming questions to PrairieLearn (July 14).
9. Stretch goals, bug fixes, tweaks, security and visual updates. (August 13).

4. Requirements

4.1 Functional Requirements

1. Create documentation for PrairieLearn deployment on docker.
2. System auto generates UML questions.
3. System auto grades UML questions.
4. Explore expanding auto generation scripts to also auto generate SQL questions.
5. Explore expanding auto grading script to also auto grade SQL questions.
6. Explore different options for more readable front-end diagram rendering.
7. Implementation of front-end diagram rendering software.
8. Canvas integration.
9. Create the ability for instructors to bulk sign up students.

4.2 Non-Functional Requirements

1. Deploy dockerized PrairieLearn.
2. Highly accurate grading system that will correctly follow set grading scheme.
3. Front-end diagram renders maintain high readability standards set by the client.
4. Compatible with Mermaid-js or nomnoml.
5. Use Python as the primary language for the back-end.
6. Extensibility and modifiability for future updates.
7. Supporting capability.

4.3 Technical Requirements

1. Questions will be randomly generated.
2. Correct autograded answer corresponds with the question.

4.4 User Requirements

4.4.1 Students User Requirements

1. Login to PrairieLearn system.
2. Users can view questions.
3. Users can answer questions.
4. Users can view feedback.
5. Users will receive a grade.
6. Users can self register.
7. Users can register via CWL login.

4.4.1 Instructors User Requirements

1. Login to PrairieLearn system.
2. Users can select specific question types.
3. Users can retrieve grades from student users.
4. Users can see a student's submission.
5. Users can set a time limit for taking the test.
6. Users can set the number of times the students can retake the test.
7. Users can bulk add accounts for students.

5. Assumptions

1. Developers will be able to program in Python 3.
2. Developers will be able to work with Docker.
3. Developers will provide weekly updates and progress.
4. Deployed instance of AutoEr with PrairieLearn on herokuapp.
5. Timely and weekly scheduled meetings with the client.
6. AutoEd autograder will be successfully imported into PrairieLearn.
7. Questions are generated automatically.
8. Questions are graded automatically.

6. High Level Risks

1. Team members getting sick for extended periods of time.
2. Team members dropping out of course.
3. Poor team synergy.
4. Not getting a prompt response from UBC IT.
5. Changes to PrairieLearn break functionality during development.
6. Unable to contact the client due to external circumstances.
7. Poor communication between developers.
8. Bad communication/relationship with client.

7. Methodology / Workflow

We will be following a test driven development cycle while also pursuing a Kanban workflow; the Kanban board will be visible and updated on GitHub Projects. As a team, we will have weekly 30min meetings (near the beginning of the week) to keep tabs on everyone's progress and to check what has been completed up to that point. We will also be holding 1 hour long meetings with our client every Thursday at 1:30pm which will allow the team to communicate progress and receive feedback on any prototyping completed during the week. Each member will use Toggle in order to keep track of how long they take on completing certain tasks, which we will later use to correct our WBS. The IDEs we selected are: Webstorm for front-end development and pyCharm for back-end development. For continuous integration we will be using Travis CI.

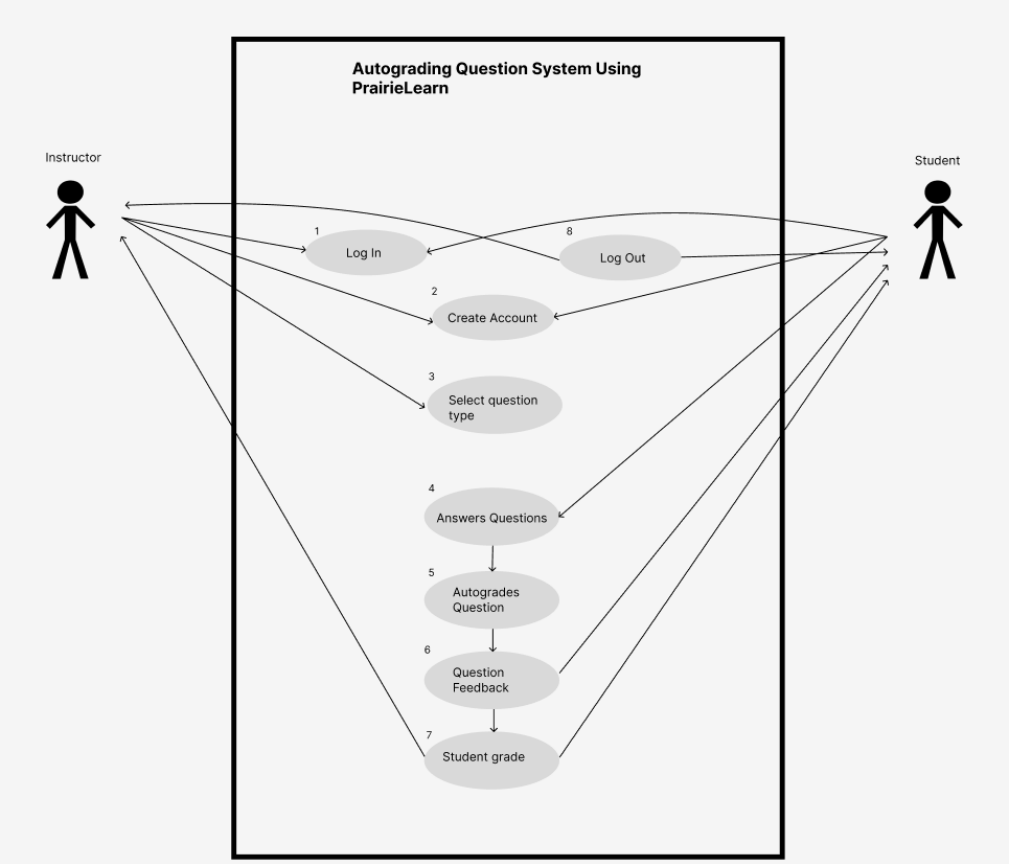
Tasks will be handled based on the WBS structure below (9). The best idea is to split off into two smaller teams and practice peer-programming. This will ensure that team members who are unfamiliar with certain work flows will be able to comfortably assimilate into the project. Testing will be written before each component and will be written to fail until said component is complete. This will ensure that the code that is written does exactly what it is supposed to do.

Canvas integration will be handled by first receiving a test course from the university, then trying the Canvas Api library to possibly get a way to integrate into Canvas. If this step is deemed to take longer than anticipated and/or unfeasible, it will be documented and terms will be renegotiated with the client.

8. UML Diagrams

The process for an Auto-grader using PrairieLearn is as follows:

An Instructor will create a generic question, which will then be stored in the database. PrairieLearn will use this generic question to create variants for students to answer. A student will then use the AutoEr front-end to answer the question and submit their answer to which PrairieLearn will automark. The mark and feedback will then be released to the student which they will be able to view.



1- Precondition- Required to get into the log in state users are required to have a stable internet condition, and an account in the system. Server must be online to log in.
Description- Users that are wanting to use the system must log into the system prior to use. This will be a basic authentication with the ability to use CWL log in credentials.
Postcondition- Once users have entered this state they will be logged in and will be required to maintain a stable internet connection to stay logged in. Once logged in they will be able to access more features.

2- Precondition- To create an account user's are required to have a stable internet condition and a valid email address for signup. Server must be online to sign up.
Description- Users that are wanting to use this system are required to create or have an account. The Create account will be an event where users can create an account using a valid email and password set up or, if under UBC, can use their CWL credentials.
Postcondition- Once a user has created an account they will be able to log into the system to start using the services. It is required that they maintain a stable internet connection to use the systems services.

3- Precondition- Users are required to have a stable internet connection to select a question type. As well their account must be linked as an instructor to access these features. The User is also required to be in the logged in state. Server must be online for the user to select a question type.
Description- This feature is for the instructor of a course to select a certain question type that will be used for the random question generation, which will then be served to the students to answer.
Postcondition- Once an instructor has set a question type, it will be set for all the students in that course. Question type can be changed at any point in the future.

4- Precondition- Users are required to have a stable internet connection to select a question type. As well their account must be linked as an student to access these features. An instructor must have already selected a question type for the student to be able to access these features. The user is also required to be in the logged in state.
Description- This feature is for the students to answer the questions the instructor has set as a question type. Questions will be shown to the students on the PrairieLearn platform where the students can answer the questions.
Postcondition- Once the student has anwered the question, the student solution will be sent to the autograding system where it will be checked for accuracy.

5- Precondition- Server must be online to complete this feature. A student must have answered a question for this feature to have become active.
Description- This feature is to take the students solution for a question and see if there solution matches the expected output.
Postcondition- Once the solution has been graded the system will go to its next feature and generate feedback for the studen.

6- Precondition- Server must be online to complete this feature. The system must have received a students answer to a question and the question must have been graded by the autograding system. The User must have a stable internet connection to receive feedback.
Description- This feature is designed to give a student feedback on the question they completed. If they got parts wrong the system will display where the user lost marks.
Postcondition- Once a user has received feedback the user must maintain a stable internet connection. The user will be able to preview there solution and read the feedback generated by the system.

7- Precondition- The server rmust be online to complete this feature. The system has to have autograded a students solution to a specific question type.
Description- The grade on the question will be stored in the system linked to the student where the student and the instructor will be able to acces the grade.
Postcondition- Once this feature is complete the student can continue to answer more questions as long as they have a stable internet connection and the server remains online.

8- Precondition- User must maintain a stable internect connection to logout. As well the server must be online.
Description- The logout is for the users to be able to leave the system and lock their account until they log in again.
Postcondition- User is not connected to the system. Accoutn is locked and can only be accessed once logged in again.

9. Work Breakdown Structure

Task List	**Average Hours**				**Average Estimate**
	Prajeet	**Emiel**	**Luis**	**Tony**	
Meetings					
Team meetings	12	12	12	12	12
Client meetings	12	12	12	12	12
Familiarise with Current Setup					
Read AutoEr and AutoEd documents	2	2	2	2	2
Setup workspace for AutoEr and AutoEd	3	3	3	3	3
Analyse and prepare current code base for import into PrairieLearn	10	10	0	0	5
PrairieLearn Set Up					
Setup IDE and workspace	1	1	3	1	1.5
Read and familiarise with PrairieLearn	4	4	4	4	4

Deploy PrairieLearn on Docker	2	3	2	2	2.25
Documentation					
Write PrairieLearn Docker Deployment document	0	3	0	0	0.75
Write Scope and Charter document	5	5	5	5	5
Write Instructor Guide document	4	5	4	4	4.25
Update and maintain Scope and Charter document	1	1	1	1	1
Kanban Board Maintenance	2	2	2	2	2
Testing					
Create mock database	0	0	1	0	0.25
Write unit tests for front-end	0	0	20	20	10
Write unit tests for back-end	20	20	0	0	10
Automate testing	3	3	0	0	0
Write unit tests for auto-grading	8	8	8	8	8

Write unit tests for auto-generation of question	8	8	8	8	8
Usability tests	2	2	2	2	2
Import UML Question to PrairieLearn					
Convert Frontend drawing from Nomnoml to Mermaid	0	0	8	8	4
Get current AutoEr to work with PrairieLearn backend	8	8	8	8	8
Get current autograding python file working with PrairieLearn	10	10	0	0	5
Create Relational Algebra Questions for PrairieLearn					
Create data structure to hold all desired Relational Algebra questions	2	2	0	0	1
Explore if Relax can be used for frontend in PrairieLearn	0	0	3	3	1.5
If not Create Frontend button layout to generate Relational Algebra Query	0	0	5	5	2.5
Create backend to pull Relational Algebra questions to properly display	7	7	2	2	4.5

Create backend to translate Relational Algebra to SQL	4	4	0	0	2
Create auto grading to compare questions expected output and queried output	5	5	0	0	2.5
Create SQL Questions for Prairie Learn					
Create data structure to hold all desired SQL questions	2	2	0	0	1
Create text box front end layout in PrairieLearn	0	0	4	4	2
Create backend to pull SQL questions to properly display	6	6	2	2	4
Create backend to query built database	6	6	0	0	3
Create auto grading to compare questions expected output and queried output					
Canvas Integration					
Contact UBC IT for help on Canvas Integration	0.5	0	0	0	0.125
PrairieLearn with AutoEr integration to Canvas	30	30	30	30	30
PrairieLearn gradebook exporting to Canvas	20	20	20	20	20

Code Review					
Github Merge requests	4	4	4	4	4
Peer code review	2	2	2	2	2
Github Repo Management	0.5	0.5	0.5	0.5	0.5
TOTAL	206	210.5	177.5	174.5	190.625
Weekly Average (12)	17.16666667	17.54166667	14.79166667	14.54166667	15.88541667

10. Approvals

Project Sponsor Signature _____

Project Manager Signature _____