Status of Software Implementation

Project 3: Building an Autograding Question System using PrairieLearn

University of British Columbia Okanagan

COSC 499 - Summer 2022

Date: August 17th, 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. Test Report

Link if not visible: [Google Spreadsheet]

Included With PrairieLean

REQUIREMENTS		Type of Test (UI: UI Testing, R: Regression Testing, I: Integrationg Testing C: Component Testing, U: Unit Testing F: Functionality Testing, SUS: System Usability Scale	Passed	Failed	Contributor (P:Prajeet Didden, E: Emiel van der Poel, L: Luis Lucio, S: Siqiao Yuan)
Functional Requirements					
1 - Student Users					
1.1	A user should be able to self register	SUS	☑	☐	
1.2	A user should be able to login to PrairieLearn System	SUS	☑	☐	
1.3	A user should be able to view questions	SUS	☑	☐	
1.4	A user should be able to answer questions	SUS	☑	☐	
1.5	A user should be able to view answer feedbacks	SUS	☑	☐	
1.6	A user should be able to receive a grade	SUS	☑	☐	
1.7	A user should be able to register via CWL login	N/A	☐	☑	
2 - Instructor Users					
2.1	A user should be able to login to Prairielearn System	SUS	☐	☐	
2.2	A user should be able to select specific question types		☑	☐	
2.3	A user should be able to retrieve grades from student users	SUS	☑	☐	
2.4	A user should be able to see a student's submission		☑	☐	
2.5	A user should be able to set a time limit for taking the test		☑	☐	
2.6	A user should be able to set the number of times the students can retake the test		☑	☐	
2.7	A user should be able to bulk add accounts for student users		☑	☐	
3 - Technical Requirements					
3.1	Questions should be able to be randomly generated.	U	☑	☐	S
3.2	Correct autograded answers should be generated corresponds with the question.	U	☑	☑	L
Non-Functional Requirements					
1 - General					
1.1	Deploy dockerized PrairieLearn		☑	☐	
1.2	Highly accurate grading system that will correctly follow set grading scheme.	U	☑	☐	L
1.3	Front-end diagram renders maintain high readability standards set by the client.		☑	☐	
1.4	Compatible with Mermaid-js or nomnoml.		☑	☐	
1.5	Use Python as the primary language for the back-end.		☑	☐	
1.6	Extensibility and modifiability for future updates.		☑	☐	
1.7	Supporting capability.		☑	☐	

2. Requirements Delivered

Included With PrairieLean

System	Status	Developer
Create documentation for PrairieLearn deployment on docker	Complete	**Prajeet**
System auto generates UML questions	Complete	**Prajeet**
System auto grades UML questions	Complete	**Emiel, Prajeet, Luis**
Implementation of front-end diagram rendering software	Complete	**Prajeet**
Canvas integration (High risk)	Incomplete	**Emiel**
Create the ability for instructors to bulk sign up students	Complete	
Marking and student grades	Complete	**Emiel**
Functionality for class/assignment creation	Complete	

System	Status	Developer
Testing	Complete	Luis, Siqiao

NFR	Status	Developer
Deploy dockerized PrairieLearn	Complete	Prajeet, Emiel
Highly accurate grading system that will correctly follow set grading scheme	Complete	Prajeet, Emiel
Front-end diagram renders maintain high readability standards set by the client	Complete	Prajeet, Emiel
Implementation of Mermaid JS	Complete	Prajeet
System can randomly generate ER questions	Complete	Prajeet
Autograder corresponds correctly to each question	Complete	Emiel

User	Status	Developer
User can Login to PrairieLearn system	Complete	Prajeet
Users can view questions	Complete	
Users can answer questions	Complete	
Users can view feedback	Complete	Emiel, Prajeet
Users receives a grade	Complete	Emiel
Users can self register	Complete	
Users can register via CWL login (N/C)	Incomplete	

Instructor	Status	Developer
Instructor can Login to PrairieLearn system	Complete	Prajeet
Users can select specific question types	Complete	

Instructor	Status	Developer
Users can retrieve grades from student users	Complete	
Users can see a student's submission	Complete	
Users can set a time limit for taking the test	Complete	
Users can set the number of times the students can retake the test	Complete	
Users can bulk add accounts for students	Complete	

Replacements/Bonus	Status	Developer
Convert UML drawing from nomnoml to Mermaid.js	Complete	**Prajeet**
Use Google oauth2 for user authentication on production server	Complete	**Prajeet**
Create custom PrairieLearn Element for UML to set specific uml questions and customise marking schema	Complete	**Prajeet, Emiel**
Prototype Relax Relational Algebra Editor tool inside PrairieLearn	Incomplete	**Team**

3. Known Bugs

Currently no known bugs and all recently discovered bugs have been fixed for the end of the project.

4. Key Lessons Learned

Time allocation into unknown tasks is hard to estimate. The team has struggled since the beginning of the project with the dockerization of PrairieLearn. More work hours should have been placed earlier onto getting the system running in production, rather than having to scramble to meet tight deadlines.

This project involved integrating an existing project into another existing project. This means reading the code of two different groups of people. It was already known to the team that it would be difficult to understand code written by other people, but we still underestimated the time it would take to get our first implementation up and running.

Team dynamic should be explored earlier to see each individual’s strengths and weaknesses, allowing the allocation of tasks to be placed favourably. This would speed up the process of development and allow the team to allocate more time to more difficult tasks.