Assignment 2: Design Document

Project 3: Building an Autograding Question System using PrairieLearn

University of British Columbia Okanagan

COSC 499 - Summer 2017

Date: June 2nd, 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 Description

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. User Groups

1. Students
2. Instructors
    a. Professors
    b. Teaching Assistants

3. Use Case Diagrams

Student User Group Use Case

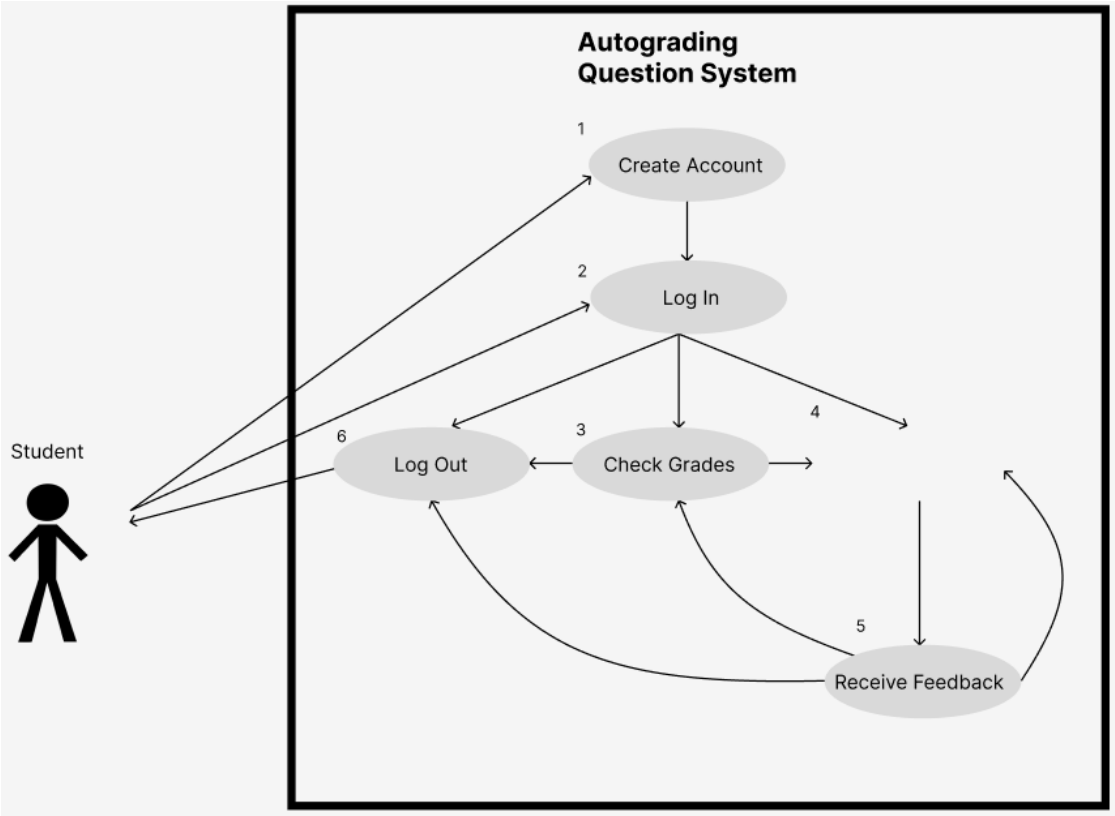

Figure 1

1- 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.*
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.*

2- 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 if the student is from UBC. Once logged in students will be capable of checking their grades, answering questions or logging out.*
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.*

3- Precondition- *Users are required to have a stable internet connection. Users are also required to be logged in. Server must be online to check grades.*
Description- *This feature is used so the student will be able to check their grades on the questions they have completed.*
Postcondition- *They will be back in a logged in state. Server must be online and the user must have a stable and reliable connection*

4- Precondition- *Users is required to maintain a stable internet connection. Users are required to be in a logged in state. The instructor of the course must have set the question type. Server must be online.*
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. The Server must remain online for this to occur and the user must have a stable internet connection.*

5- 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.*

6- 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. Account is locked and can only be accessed once logged in again.*

Instructure User Group Use Case

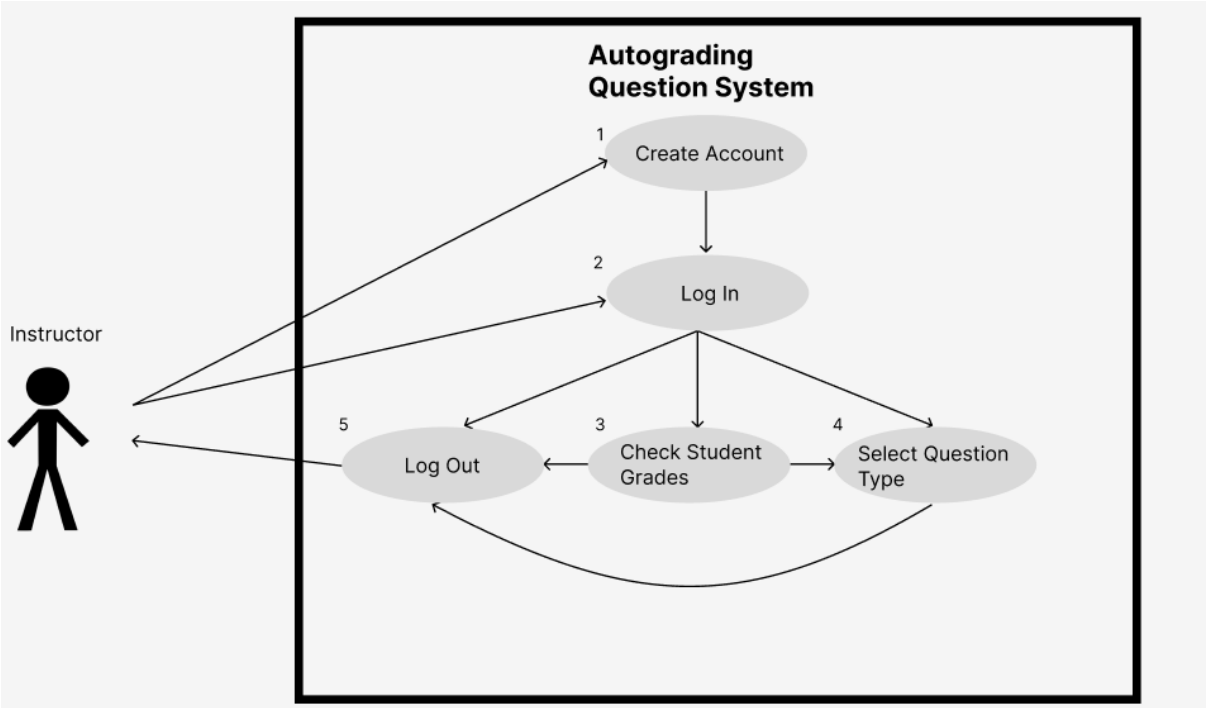

Figure 2

1- 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. Instructors also will be able to bulk add accounts for all students in their courses.*
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.*

2- 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. Once logged in instructors will be capable of checking student grades, Setting a question type, and to log out.*
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.*

3- Precondition- *Users are required to have a stable internet connection. Users are also required to be logged in. Server must be online to check student grades.*
Description- *This feature is used for instructors to see how their students are doing in the classes. They will be able to see individual student grades.*
Postcondition-

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 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 a user 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.*

5- 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. Account is locked and can only be accessed once logged in again.*

4. System Architecture

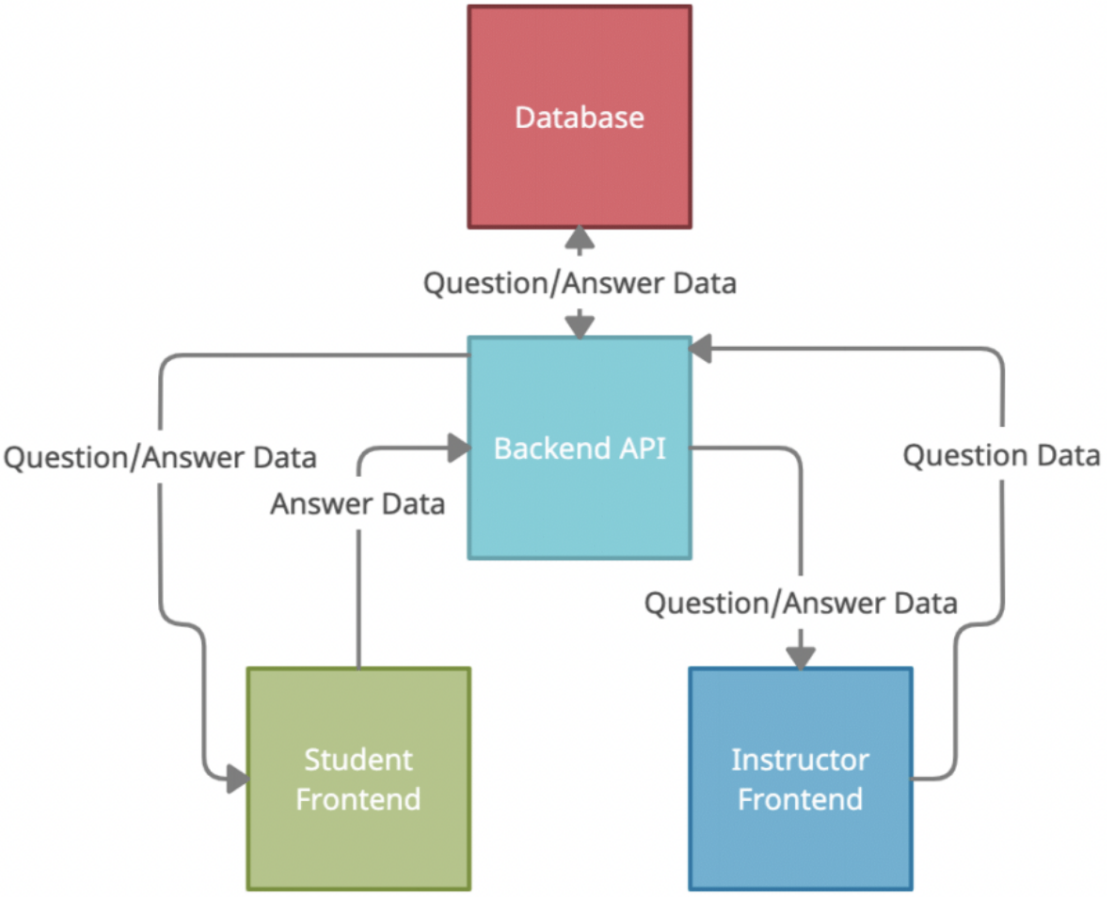

Figure 3: *The instructor frontend will allow users with instructor privileges to enter question data and send that information to the backend API. The student frontend will give users questions generated by the instructor, which they will be able to answer through the use of the frontend interface in order to also send that data to the backend API. The backend API will receive both question templates and individual student answers, and will store all of this data on a database that is capable of linking each answer to its corresponding student. This API will be responsible for both communicating with the database as well as sending/receiving all data to corresponding frontend interfaces.*

5. UI Mockups

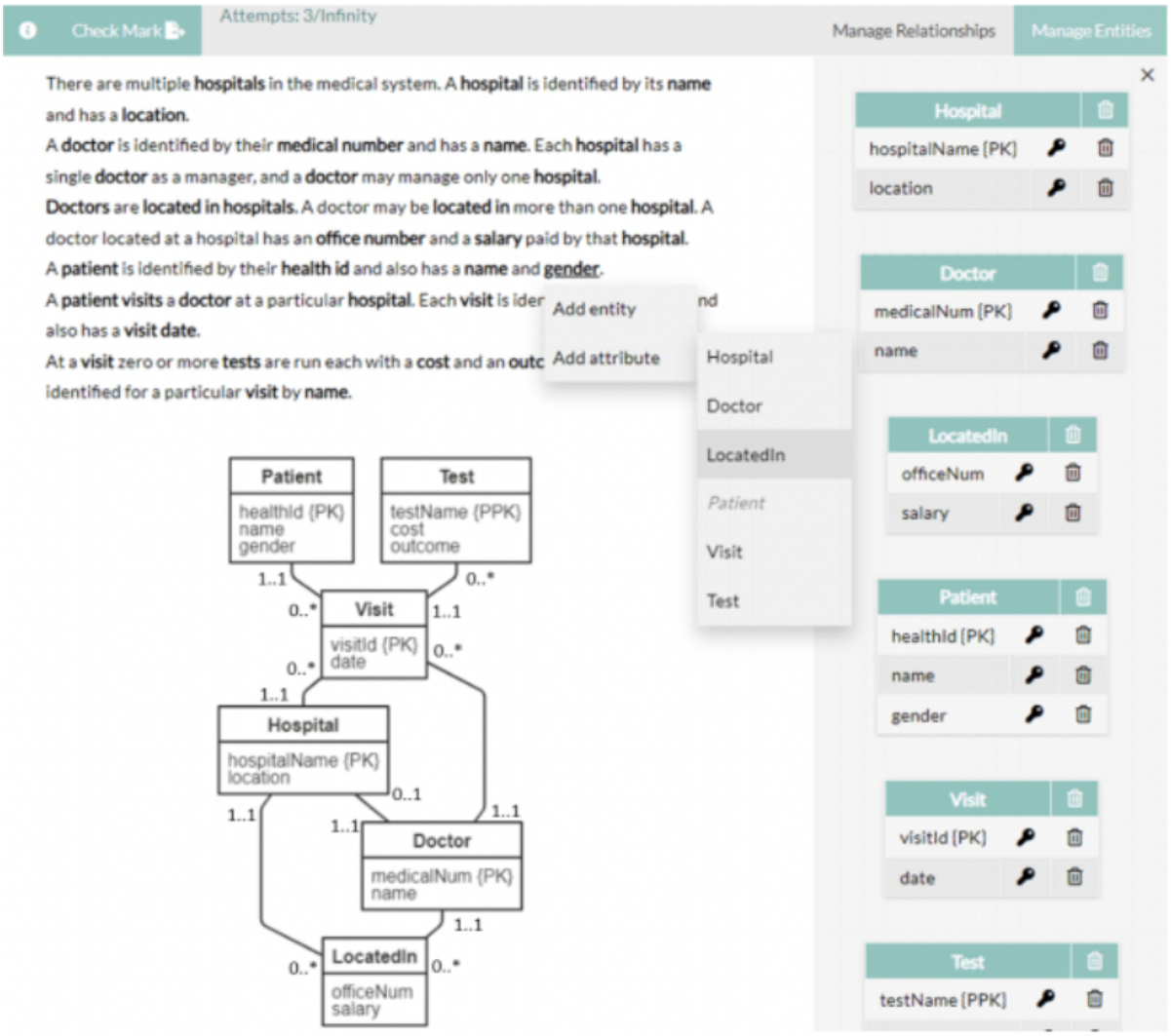

Figure 4: *This figure showcases the way the student can interact with the question's interface in order to generate a UML diagram. The randomly generated question will have several keywords written in bold, which the student can click in order to add that word to the diagram either as a new entity, or an attribute of an already existing entity. The options shown in the top right corner will allow students to manage already existing entities by selecting certain attributes to be primary keys, or delete them in case they made a mistake. The Manage Relationships menu allows students to link entities to each other and form a relationship between them, where they will also be allowed to specify the cardinality of the relationship. After the students finish the question, they can click Check Mark in order to receive a grade on their submission.*

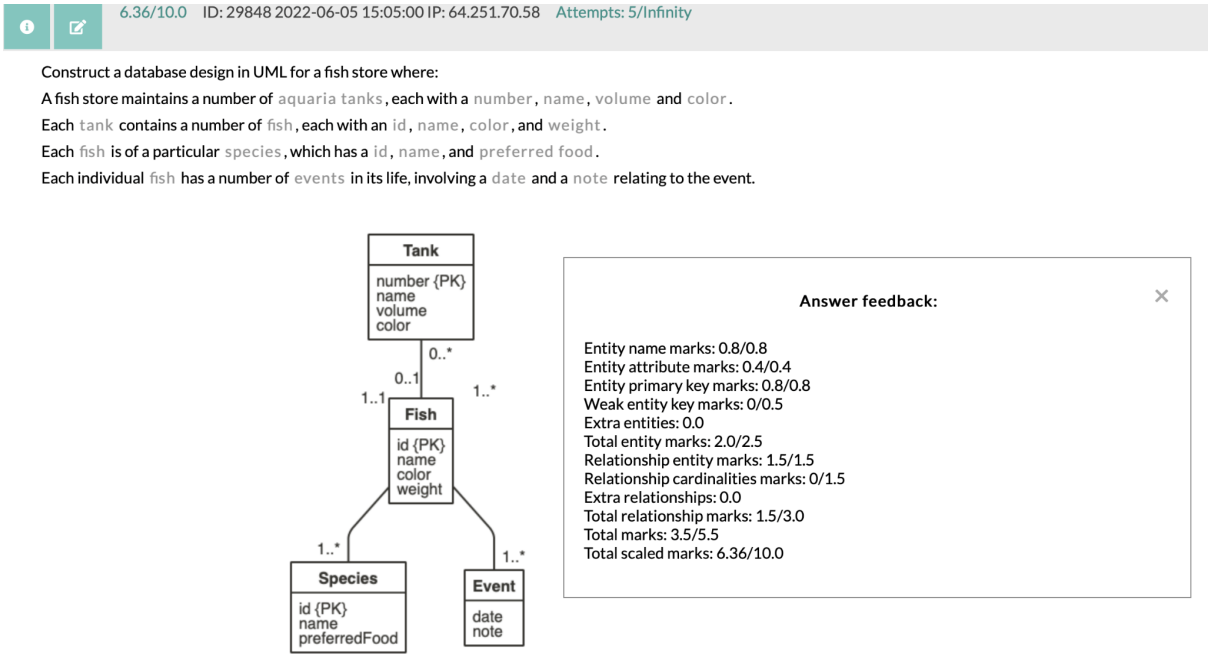

Figure 5: *This figure showcases the grading and feedback portion of the interface. After attempting the question, the students can Check Mark in order to receive marks for their answer and check their results. After confirming the marks awarded, students will also be able to look at the specific marking scheme by clicking the green highlighted grade and receive feedback in order to know exactly where they got their marks from and where they lost marks in the case they made a mistake.*

6. Tools To Build Software

1. Frontend IDE - WebStorm.
2. Backend IDE - PyCharm.
3. Time Tracking - Toggl.
4. Task Tracking - Github Projects.
5. Continuous Integration - Travis CI.

7. Test Plan

7.1 Unit Testing

- Unit testing will be used to verify that functions results are as expected. Unit testing will be used throughout the whole project to help build up functions.
- Coverage testing should apply to all code in the codebase we have written. This will allow us to see which lines of code are being used which in turn can help optimise code.

7.2 Regression Testing

- Swapping the backend from Autoed to PrairieLearn will require regression testing, which can be easily implemented by using the tests written for Autoed with minor modifications. An example test could be checking whether the autograding script still works on the sample questions from the preceding project. Using the tests from the previous project and checking their output, then making sure the output of the modified system is the same will be the way to check if the system responds in the same way.

7.3 Integration Testing

- System integration testing will be written for connecting Autoer and PrairieLearn components. A test could be written to check whether data accessibility is happening correctly.

7.4 Component Testing

- Functionality testing between components, to ensure that they work together. This will test over classes and different files. These tests can be written on a per component basis, and should fail before writing each component.

7.5 User Testing

- User login will be tested for correctness of credentials, empty fields, incorrect credentials, and if the user is logged in successfully or not. This can be tested by creating tests with incorrect credentials, and tests with correct credentials.
- Logout will be tested for if the user is logged out once the logout button is clicked.
- User creation will be tested for field validation, so each field is filled out with the correct type. Testing can be done to check if the email field is only taking emails, Name field does not accept numbers and symbols, date fields only take numbers and months, and password field matches some form of security requirements.
- Uploading user credentials can be tested to see if the excel file is being accessed, user credentials are being pulled correctly, and if the table is populating correctly.

7.6 Functionality Testing

- The functionality of the system is based on whether or not the system can accurately grade UML questions in the way it did before the shift to PrairieLearn. Testing can be put in place to check if the question is being graded, by checking if when entities are clicked, they are updated on the grading scheme. The grading scheme should align with the original system (AutoEd).

8. References

1. *Foss, S., Urazova, T., & Lawrence, R. (n.d.). Automatic Generation and Marking of UML Database Design Diagrams.*
2. *Urazova, T. (2022). Building a System for Automated Question Generation and Evaluation to Assist Students Learning UML Database Design. [Honours Thesis, University of British Columbia].*