Automating Database Question Generation and Marking with PrairieLearn

Handover Documentation

Version:
First Version - July 28th, 2023
Final Version - August 15th, 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. Code Location

Code repository link:
https://github.com/UBCO-COSC-499-Summer-2023/project-3-a-automating-database-question-generation-automating-db-q-gen-marking-team-a

2. RelaxQueryApi Location

RelaXAPI repository link (**required for grading Relational Algebra questions**):
https://github.com/azipis/RelaXQueryAPI

3. Deploy Location

Course repository link: https://github.com/PrairieLearnUBCO/pl-ubco-cosc304

4. Development Installation Dependencies

- Python (version 3.11.3)
- Docker (version 4.21.1)
- Github
- Visual Studio Code (or another IDE)
- Internet browser with development tools (Chrome / Firefox)
- Github Desktop (Optional)
- Docker Desktop (Optional)

5. Development Instructions

Docker Documentation for PrairieLearn

1. Install docker, docker-compose, and docker-desktop(optional)
2. git clone this repository: `https://github.com/UBCO-COSC-499-Summer-2023/project-3-a-automating-database-question-generation-automating-db-q-gen-marking-team-a.git`
3. In your terminal, navigate to cloned directory.
4. In your terminal run `docker-compose up`
5. To check if the docker container is running, run `docker-compose ps`. if the response is empty, run `docker-compose ps -a`, and it will show the status of the closed docker container.

Creating the relaxAPI container

1. Visit repository found at `https://github.com/azipis/RelaXQueryAPI`
2. Follow directions found at that repository.

6. Deployment Installation Dependencies

- Docker (version 4.21.1)
- Github
- Internet browser

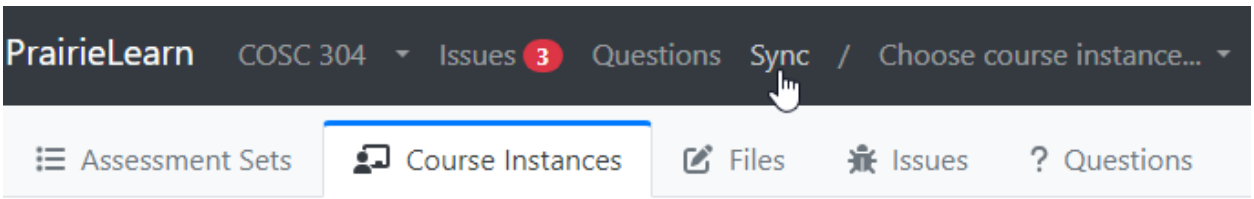
7. Deployment Instructions

7.1. Deploy PrairieLearn using the instructions found here: https://prairielearn.readthedocs.io/en/latest/running-in-production/setup/

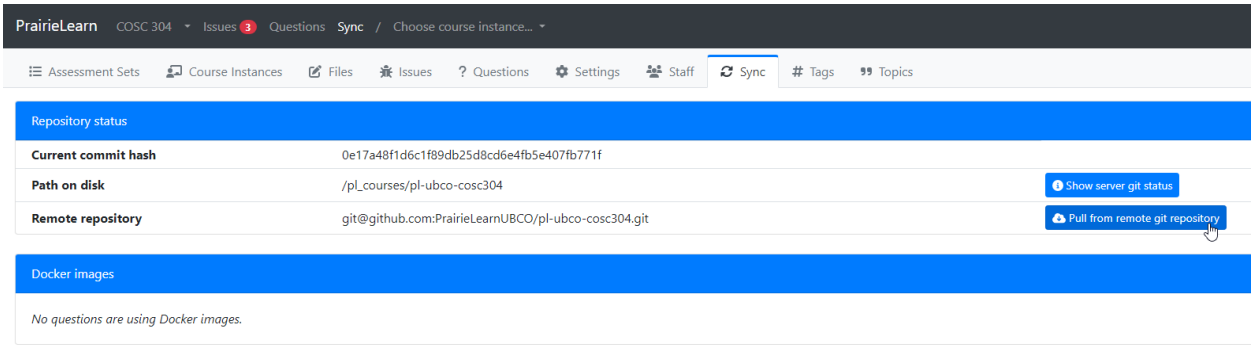
7.2. Push all course changes (elements, questions, additional libraries) to course repository (https://github.com/PrairieLearnUBCO/pl-ubco-cosc304)

7.3. Login to PrairieLearn as an admin/staff user.

7.4. Click sync in the navbar:



7.5. Pull from git course repository:



8. Testing Instructions

8.1. Unit

Install Unittest. There are python unit tests in the sql-element, relax-element, and in the serverFilesCourse/RASQLib folders. Simply navigate to these folders in your terminal and run python -m unittest.

8.2. UI / Integration

Install Selenium. Start the instance on localhost:3000. All the UI/Integration tests are in the tests/ui folder. Once you are there, the UI test files are all in the ddlQuestions, relaxQuestions, and sqlQuestions folders. You need to run python -m unittest in each folder to run all tests.

8.3. Performance

Install Locust. Go to tests/performance. There you have folders RelaX and SQL. In each folder there are 2 subfolders with the test files in them. The performance test files are all named “locustfile.py”. When you’re in a folder, type locust in your terminal and open up the localhost server that the locust interface is on to run your tests with varying loads.

9. Testing Not Implemented

JavaScript unit testing was not implemented for our renderer.js file due to the difficulty of unit testing JavaScript containing jQuery. To clarify, we were not able to figure out a way to mock DOM elements that are required for our nested functions that used a ready/loaded document. With that being said, the functions of these files are being tested through our UI and Integration tests which would fail if our renderer file contained issues.

10. Features Not Implemented

This section is not applicable. All requested features were implemented, working and deployed on live servers.

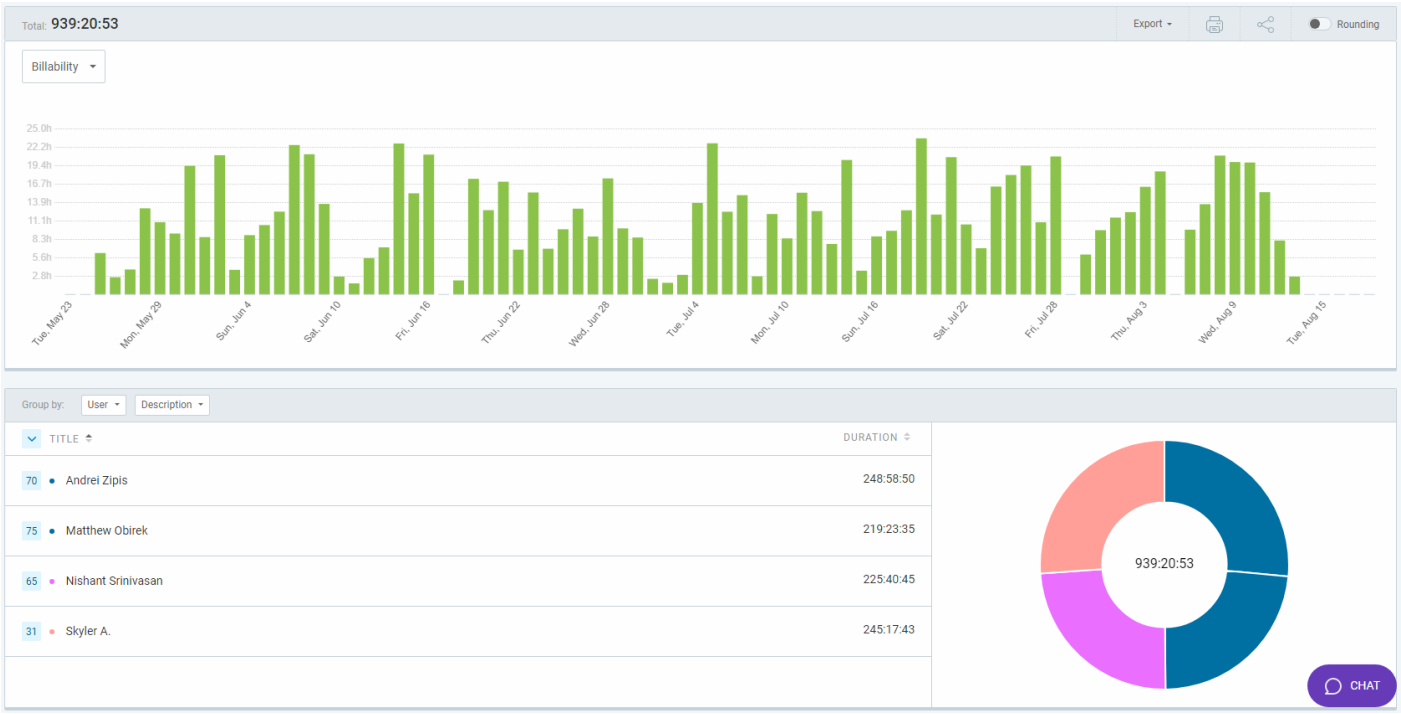
11. Future Work

As our group fully implemented the first three labs of Intro into Databases (*COSC 304*) - Relational Algebra, SQL table creation, and SQL queries - and the previous years group implemented labs 4 and 5 - Database design and UML modeling, and converting UML diagrams to Relational models - Future work with this project could include many different things. We recommend these future projects:

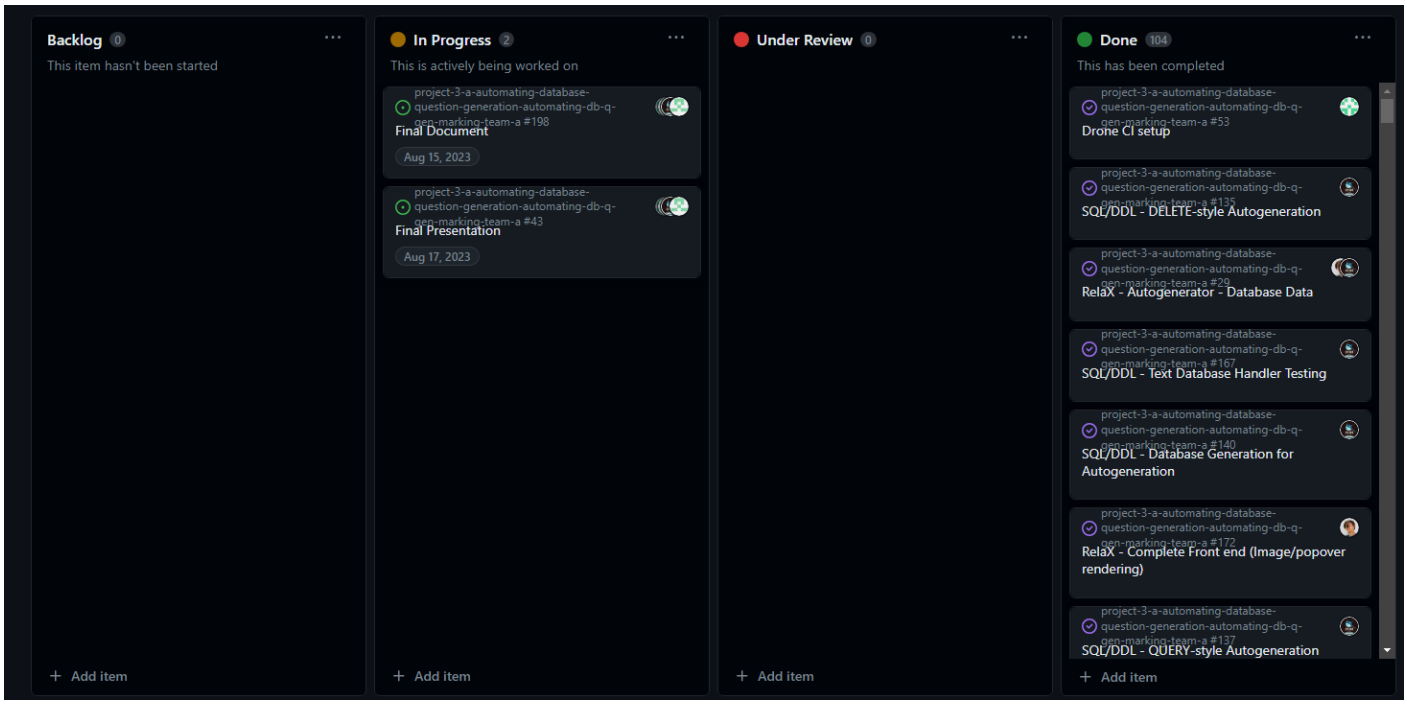
a. Groups could implement future labs on XML, JSON, Views, and triggers. We recommend using the codemirror library, as it is what we used to implement our SQL and RelaX front-end editors.
b. Groups could utilize machine learning, or AI, to improve the grammatical and semantic accuracy of the Automatic generation for the question text and the database layout.
c. Groups could also work on improving performance. We recommend that if the RelaX autograding remains a bottleneck, rewriting the executeRelalg function, and the Relation object, from the Relalg_bundle.js in python using this python library - https://pypi.org/project/dbis-relational-algebra/. The library is built by the same developer of the relax editor. Alternatively, if a faster Relalg_bundle.js is created - one that executes faster - that would solve the performance issue. However, we suggest rewriting it in python because that would reduce the roughly 3 second communication delay between relaxAPI and Prairielearn, and would potentially increase the execution speed of the queries.
d. As we have not tested our deployment with PrairieLearn Team B's gamification scoreboard, we do not know if there are any conflicts with our projects. Testing, issue resolution, and integration of past projects could be a project in itself, if there happen to be softwear breaking bugs.

12. Workflow Statistics

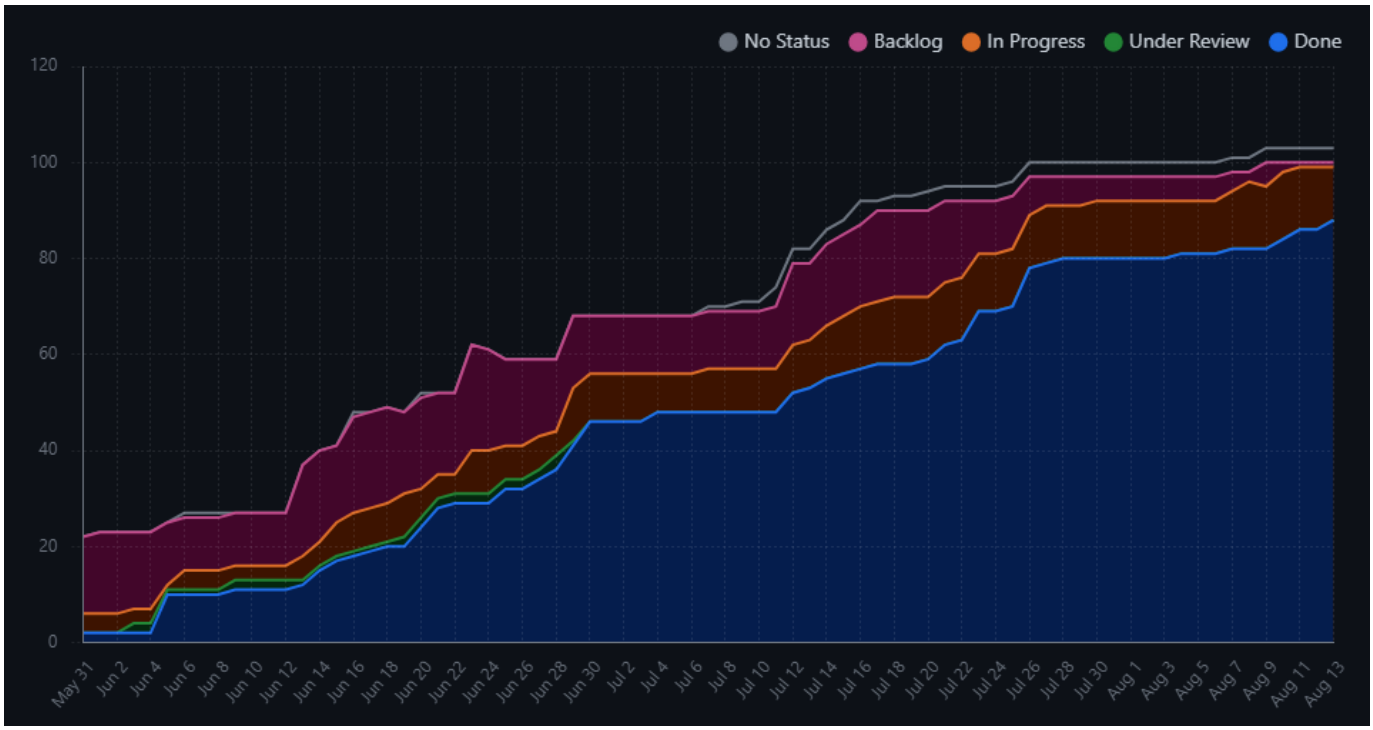
12.1. Clockify Hours



12.2. Kanban Board



12.3. Burnup Chart



Note: Github error in their chart generation. At this point only two issues were still in progress.