Summary

The motivation behind the CSCE 625 Self Evaluation project is to help students in assessing their preparation for the graduate-level Artificial Intelligence course. This project can assist students in understanding the course prerequisites. By using this evaluation portal, the students can make an informed decision as to whether they are sufficiently prepared to register for the class.

The requirements for the project were provided by Dr Duncan Walker, Professor, Texas A&M University. The key stakeholders of this project are prospective students, teaching assistants, and the course instructors.

As this was a legacy application, a majority of the work dealt with enhancing existing features or implementing architecture/design fixes. The key customer requirements were:

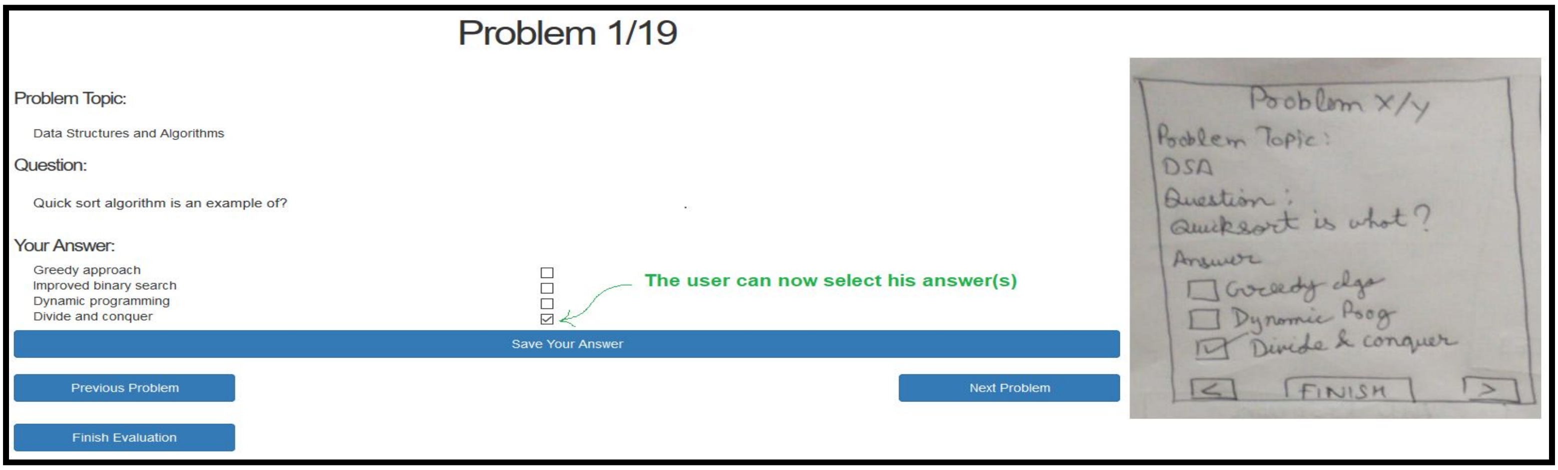
1. Implementing Multiple-Choice Questions(MCQs)
2. Supporting Mathematical equations in questions
3. Automating grading/evaluation
4. Setting up a password recovery system

Through this project, all the above requirements were met. The portal now supports Multiple-Choice Questions on top of the basic “Short Answer” questions. The feature to support Mathematical equations in Questions/Answers was done using the MathJax library. Automatic grading has been implemented for both MCQ and Short Answer questions in the Quiz module. In addition, the instructors now have the opportunity to reset “Lost Password”s.

Important User Stories

The team worked in an Agile fashion to meet the customer requirements. Hence, the requirements were broken into User Stories which could be completed in two-week iterations. Most of the user stories were assigned 1 points because developer usually had full 2 weeks to work on them and other team members were always available to jump in to help. The user stories that got assigned 2 points usually involved a lot more coding (hence testing) or had less time available to be completed. Following are few of important User Stories:

- User Story #1 (2 points):
  Modify the views(questions) according to the new DB schema
  As a user, I should be able to check my choice of answer for multiple choice questions. This story was implemented in Iteration 2, and it allowed a student to select his choice of answer(s) using check-boxes provided for a multiple-choice question complementing the database changes done to store the options in a Options table.



Problem 1/19

Problem Topic:

Data Structures and Algorithms

Question:

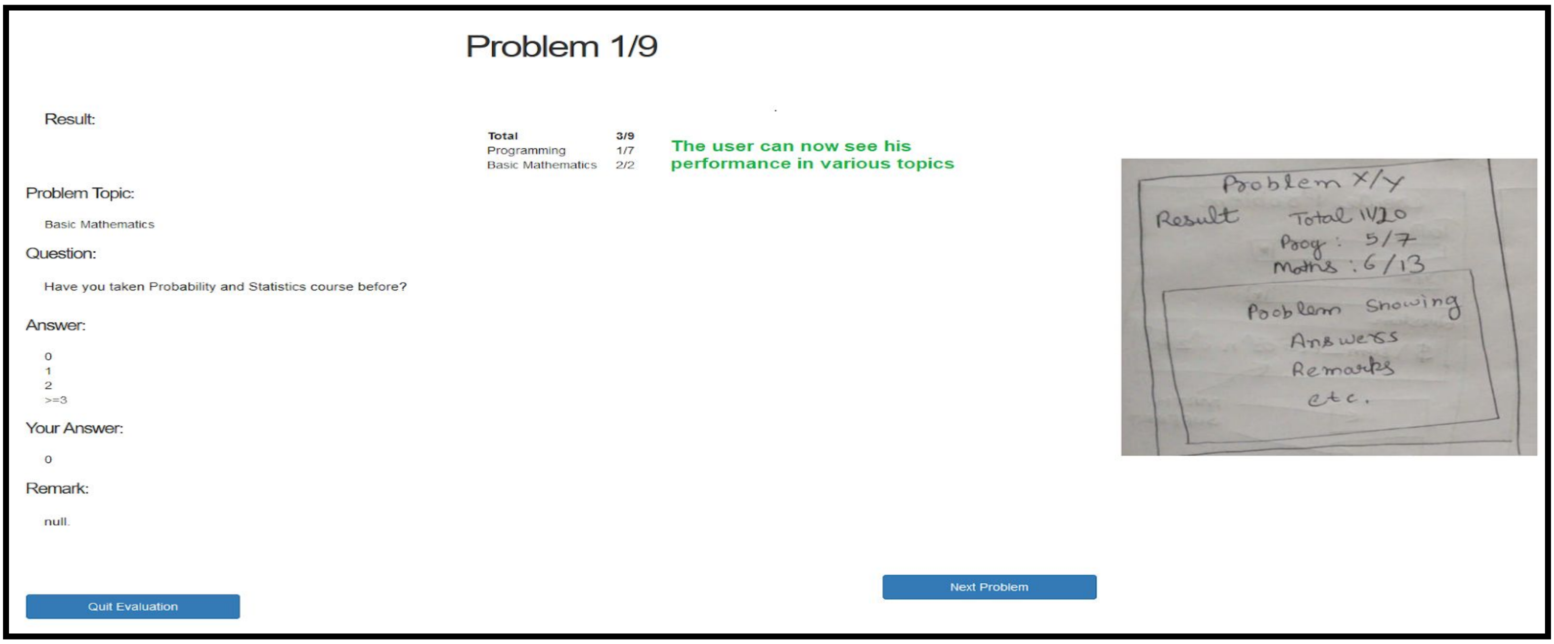
Quick sort algorithm is an example of?

Your Answer:

Greedy approach □
Improved binary search □
Dynamic programming □
Divide and conquer ☑

The user can now select his answer(s)

- User Story #2 (1 point):
  Add the logic for the calculation of score/report
  As a user, I want to check my performance in various topics after finishing my evaluation.
  This story was implemented in Iteration 2, and it allowed a student to view his score in every topic once he finishes the evaluation.



Problem 1/9

Result:

Total	3/9
Programming	1/7
Basic Mathematics	2/2

The user can now see his performance in various topics

Problem Topic:

Basic Mathematics

Question:

Have you taken Probability and Statistics course before?

Answer:

0
1
2
>=3

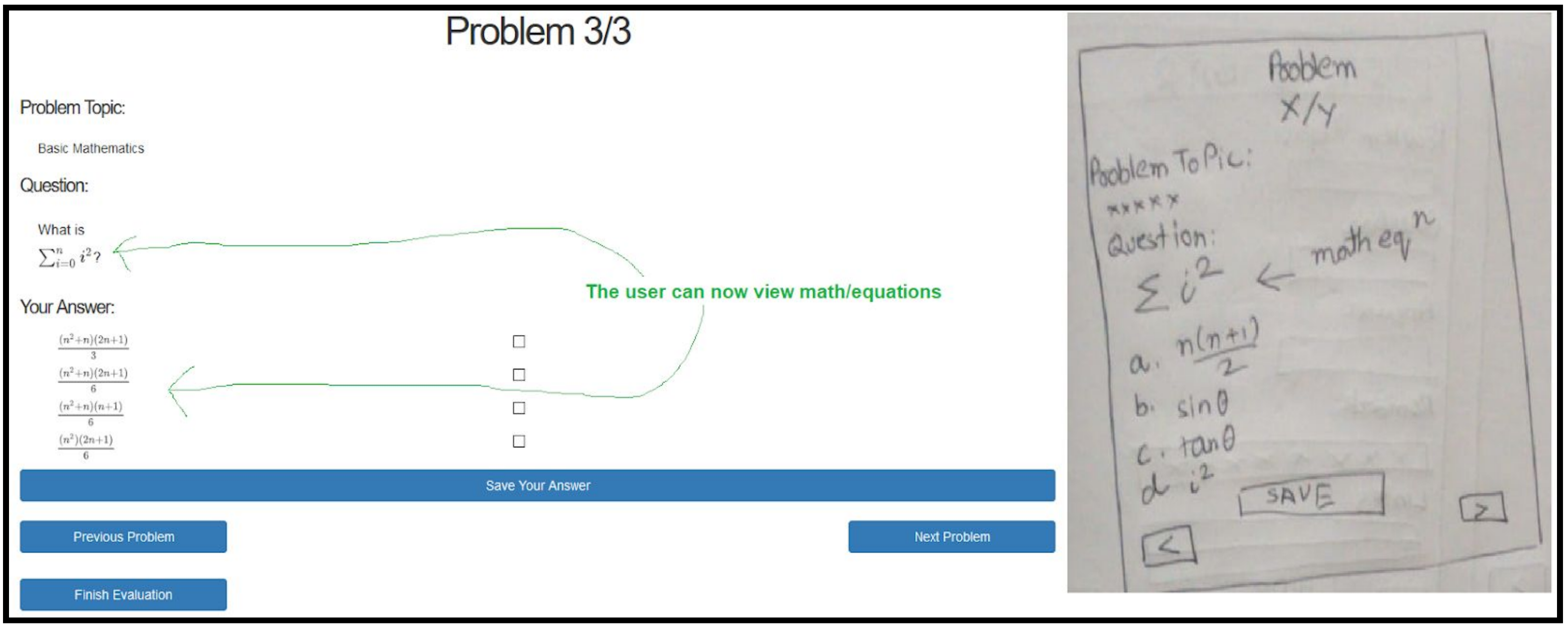
Your Answer:

0

Remark:

null.

- User Story #3 (1 point):
  Support mathematical equations
  As an instructor, I should be able to add mathematical equations in my questions/answers/choices.
  This story was implemented in Iteration 2, and it gave the Instructor the flexibility to add mathematical equations in question, answer and option fields. Also allowed the user to view math/equations related questions.



Problem 3/3

Problem Topic:

Basic Mathematics

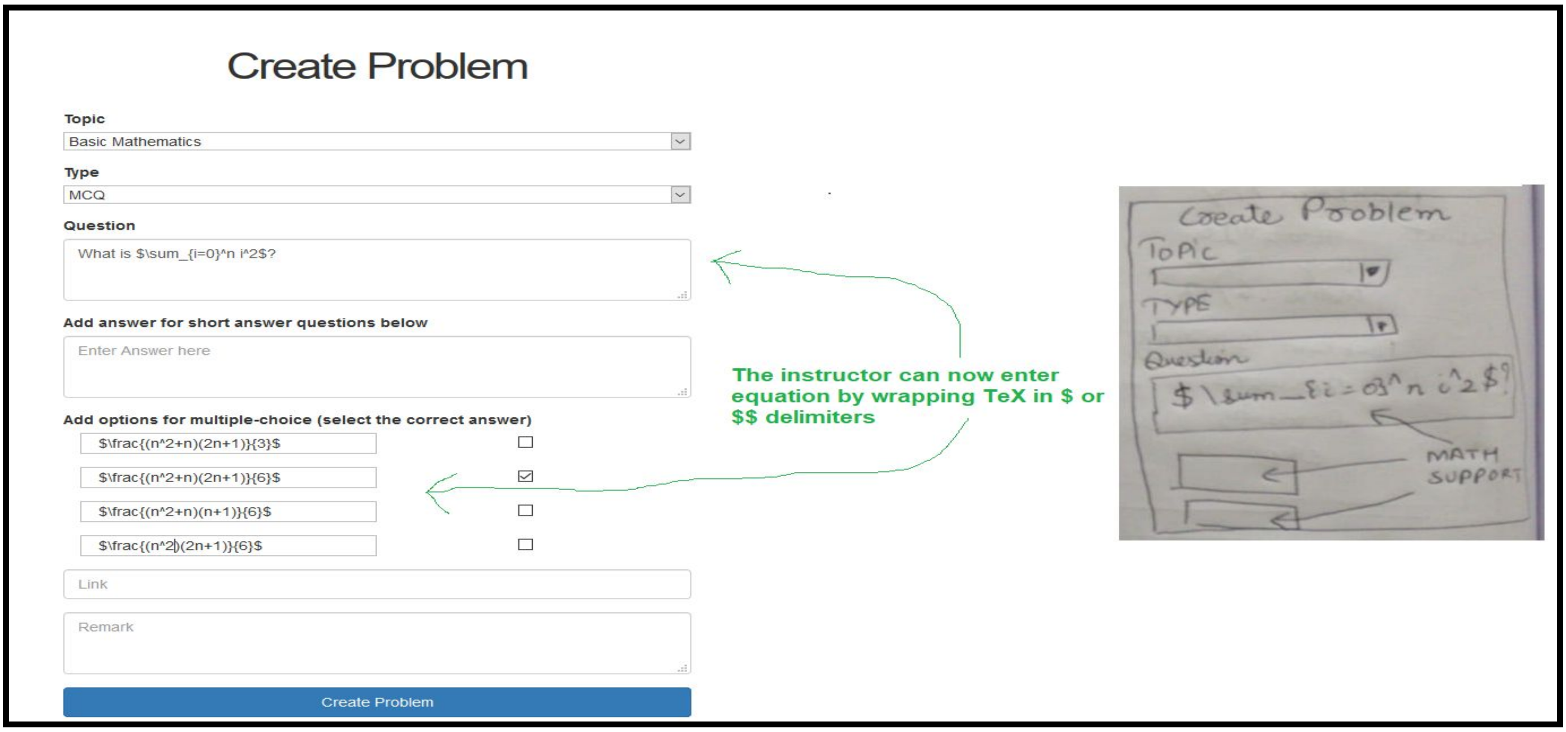
Question:

What is $\sum_{i=0}^{n} i^2$?

The user can now view math/equations

Your Answer:

$\frac{(n^2+n)(2n+1)}{3}$ □
$\frac{(n^2+n)(2n+1)}{6}$ □
$\frac{(n^2+n)(n+1)}{6}$ □
$\frac{(n^2)(2n+1)}{6}$ □



Create Problem

Topic

Basic Mathematics

Type

MCQ

Question

What is $\sum_{i=0}^n i^2$?

Add answer for short answer questions below

Enter Answer here

Add options for multiple-choice (select the correct answer)

$\frac{(n^2+n)(2n+1)}{3}$	□
$\frac{(n^2+n)(2n+1)}{6}$	☑
$\frac{(n^2+n)(n+1)}{6}$	□
$\frac{(n^2)(2n+1)}{6}$	□

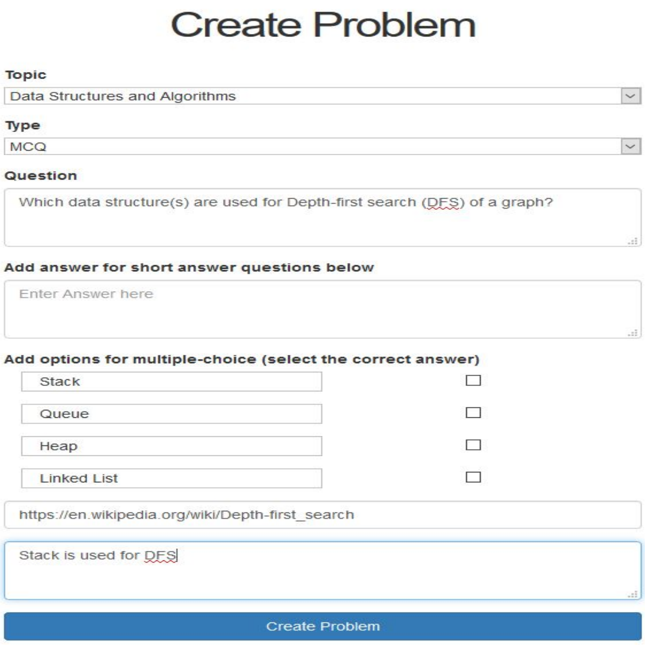
Link

Remark

The instructor can now enter equation by wrapping TeX in $ or $$ delimiters

- User Story #4 (1 point):
  Password recovery system
  As a user, I should be able to recover my passwords.
  This story was implemented in Iteration 2, and a forgot password option has been added for the users to be able to recover their password. The reset link for the password will be sent to the email address used during registration.

- User Story #5 (1 point):
  Change the add question interface according to the new DB schema
  As an instructor, I should be able to add questions of type both Multiple choice and short answer.
  This story was implemented in Iteration 2, and with this, the Instructor was able to add both Multiple choice and short answer type questions. For the short answer type questions, the correct answer has to be entered and for the multiple choice questions, Options along with a correct option has to be entered.



Create Problem

Topic

Data Structures and Algorithms

Type

MCQ

Question

Which data structure(s) are used for Depth-first search (DFS) of a graph?

Add answer for short answer questions below

Enter Answer here

Add options for multiple-choice (select the correct answer)

Stack	□
Queue	□
Heap	□
Linked List	□

https://en.wikipedia.org/wiki/Depth-first_search

Stack is used for DFS

The instructor can now

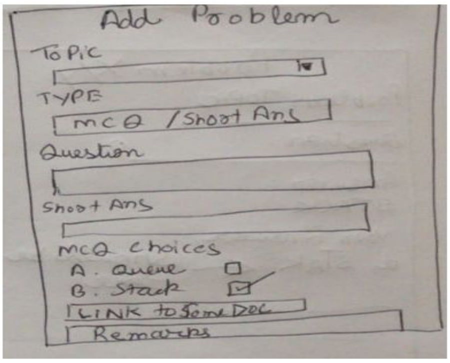
choose the type of question - multiple-choice or short-answer

add the answer in case the question is of type 'short-answer'

add choices for a multiple-choice question

choose the right choice(s)

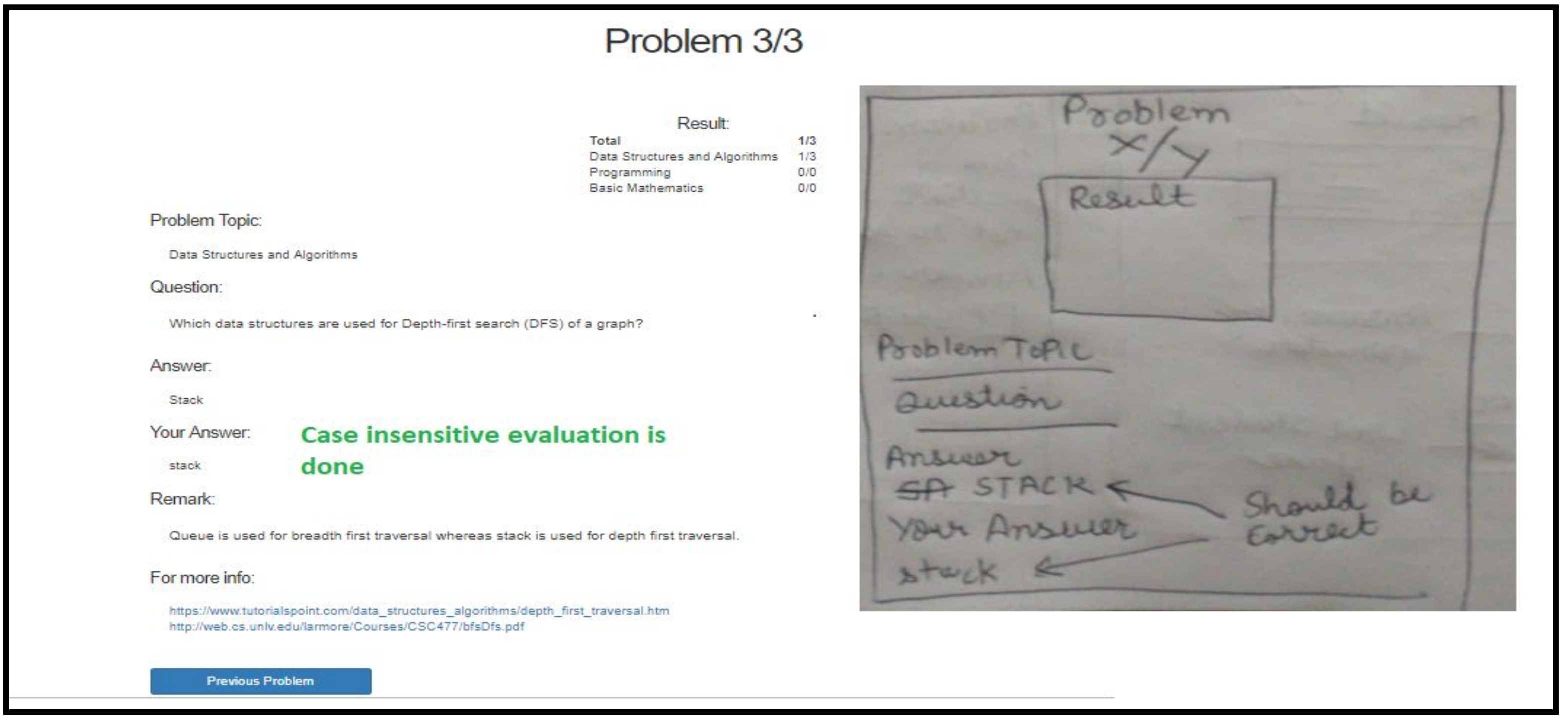
add a link in the answer's explanation



- User Story #6 (1 point):
  Add link(s) in the answer's explanation
  As an Instructor, I should be able to provide URLs to external sources to the students
  This story was implemented in Iteration 2, and it facilitated the Instructor to provide one or more links to external websites for the students to refer for further explanation on the questions.

- User Story #7 (1 point):
  Logic for Short Answers
  As a user, I should be able to see my answers for short answer questions evaluated.
  This story was implemented in Iteration 3, and logic for evaluating short answer questions was added as a part of it. A case-insensitive exact string match has been used to evaluate the short answer questions.

- User Story #8 (1 point):
  Consider the short answer in performance report
  As a user, I should be able to see my performance in short answers on the evaluation report.
  This story was implemented in Iteration 3, and after the logic for evaluating short answers has been added in this iteration as a part of User Story #8, the feature for including this evaluation report in the overall score report has also been added. With this, now the user will be able to see his overall performance inclusive of both short answers and MCQs at the end of evaluation.



Problem 3/3

Result:

Total	1/3
Data Structures and Algorithms	1/3
Programming	0/0
Basic Mathematics	0/0

Problem Topic:

Data Structures and Algorithms

Question:

Which data structures are used for Depth-first search (DFS) of a graph?

Answer:

Stack

Your Answer:

stack

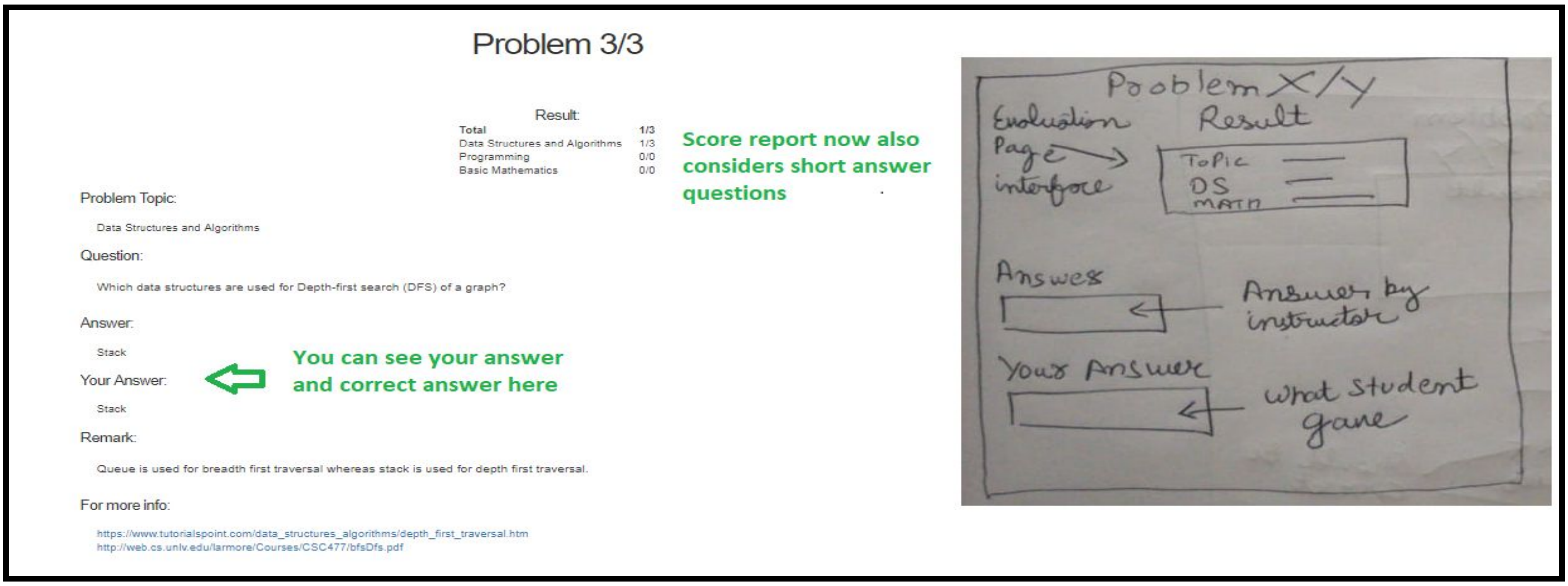
Case insensitive evaluation is done

Remark:

Queue is used for breadth first traversal whereas stack is used for depth first traversal.

For more info:

https://www.tutorialspoint.com/data_structures_algorithms/depth_first_traversal.htm
http://web.cs.unlv.edu/larmore/Courses/CSC477/bfsDfs.pdf



Problem 3/3

Result:

Total	1/3
Data Structures and Algorithms	1/3
Programming	0/0
Basic Mathematics	0/0

Score report now also considers short answer questions

Problem Topic:

Data Structures and Algorithms

Question:

Which data structures are used for Depth-first search (DFS) of a graph?

Answer:

Stack

Your Answer:

Stack

You can see your answer and correct answer here

Remark:

Queue is used for breadth first traversal whereas stack is used for depth first traversal.

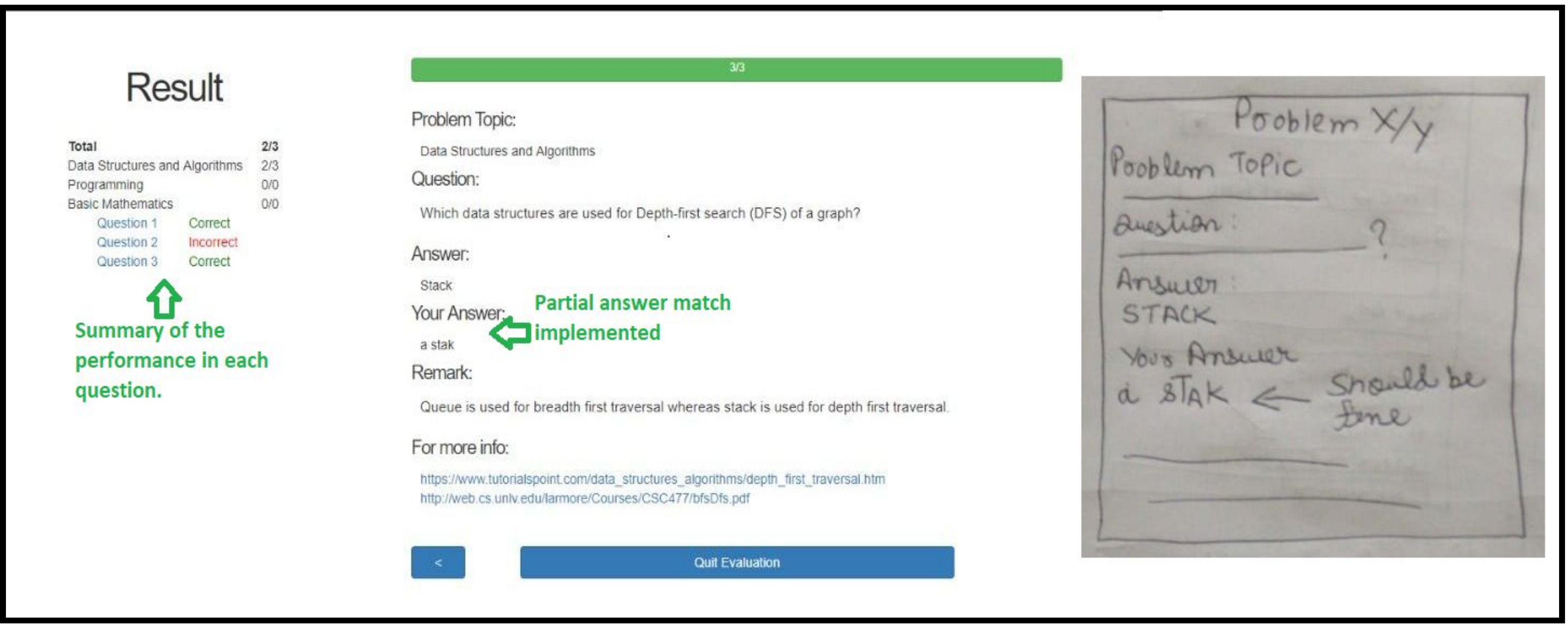
For more info:

https://www.tutorialspoint.com/data_structures_algorithms/depth_first_traversal.htm
http://web.cs.unlv.edu/larmore/Courses/CSC477/bfsDfs.pdf

- User Story #9 (2 points):
  Include the code for partial checking of the short answer questions
  As a user, I should be able to see my short answers evaluated even if there's a partial answer.
  This story was implemented in Iteration 4. Earlier, an exact match was used to evaluate the short answer questions in Iteration 3, but in this iteration, functionality has been added that can check partial answers and effectively map them to the correct answer for any question. Cases like minor spelling errors, extra/missing articles have been handled.

- User Story #10 (1 point):
  Summarize the performance report results on the UI
  As a user, I should be able to see a detailed performance report once I finish my evaluation

This story was implemented in Iteration 4, and it was an extension to the User Story #2, which included displaying the performance report. In addition, the summary of whether or not every question is correct is displayed. Also, functionality to access the desired question directly has been added.



Result

Total	2/3
Data Structures and Algorithms	2/3
Programming	0/0
Basic Mathematics	0/0
Question 1	Correct
Question 2	Incorrect
Question 3	Correct

Summary of the performance in each question.

3/3

Problem Topic:

Data Structures and Algorithms

Question:

Which data structures are used for Depth-first search (DFS) of a graph?

Answer:

Stack

Your Answer:

a stak

Partial answer match implemented

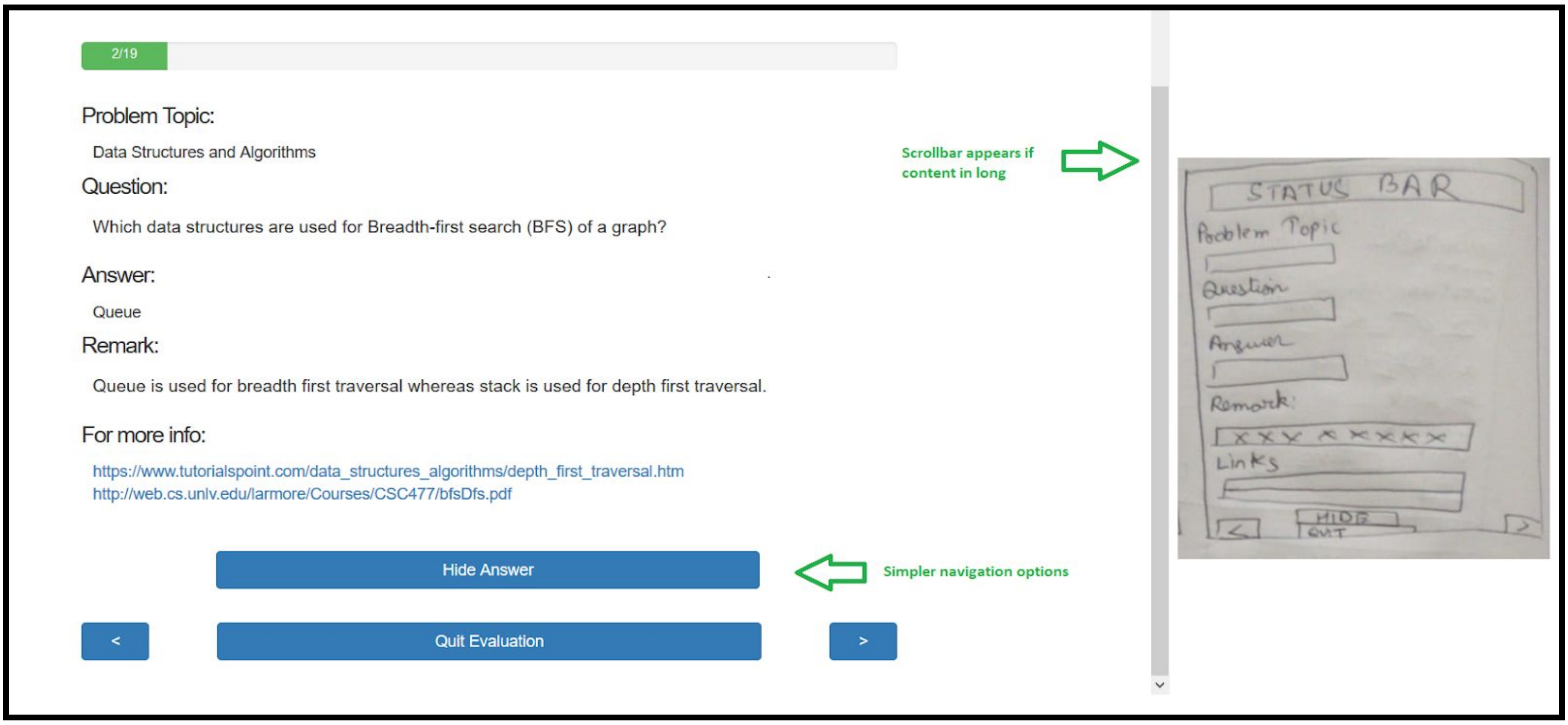
Remark:

Queue is used for breadth first traversal whereas stack is used for depth first traversal.

For more info:

https://www.tutorialspoint.com/data_structures_algorithms/depth_first_traversal.htm
http://web.cs.unlv.edu/larmore/Courses/CSC477/bfsDfs.pdf

- User Story #11 (1 point):
  Modernize the question UI and evaluation UI
  As a user, I would like to navigate the Quiz and Flashcard modules comfortably.
  This story was implemented in Iteration 4. As a result of this story, the navigation between the questions and the evaluation were made simpler. Also, the website was made responsive by restricting the contents to the window size(a scrollbar will appear in case the contents overflow).



2/19

Problem Topic:

Data Structures and Algorithms

Question:

Which data structures are used for Breadth-first search (BFS) of a graph?

Answer:

Queue

Remark:

Queue is used for breadth first traversal whereas stack is used for depth first traversal.

For more info:

https://www.tutorialspoint.com/data_structures_algorithms/depth_first_traversal.htm
http://web.cs.unlv.edu/larmore/Courses/CSC477/bfsDfs.pdf

Scrollbar appears if content in long

Simpler navigation options

Legacy Product Understanding

To understand the previous teams work, we went through their report and also checked the version of the web app that they deployed on Heroku. We found out that their database design is not very well suited to our user stories. For example, their DB design used to store 4 options per question in the same table. Also, there was no question type column which could differentiate the MCQ questions from short answer type questions. As a result, upon discussion with our customer, we decided to make changes to the DB design even before we start implementing any user stories. We moved options to a completely different table allowing us to have a variable number of options per question and also created a separate "question type" table that allowed us to implement performance evaluation user story later. The time spent on careful DB design really helped us in later iterations. Secondly, the previous team did implement the user registration functionality but there was no mechanism to recover or change the password. We discussed among us and decided to implement that functionality upon discussion with the customer. One major issue that limited our flexibility for UI changes was the last team's decision to use ".erb" files for view pages instead of our preferred ".haml" files. However, we decided to live with that decision and continue to make changes to "erb" files itself because making the corresponding "haml" changes would have taken a lot of our effort and it was in our best interest to just learn "erb" syntax as and when required.

Team Roles

Aditya Kumar was the Product Owner for the entirety of the project. Yashwanth Reddy, Akhila Mangipudi, and Mayukh Roy Chowdhury acted as Scrum Masters alternately. Development and Testing tasks were taken up by all six members of the team.

Summary of Iterations

Iteration 0: Met the customer to understand his expectations and for formulating the requirements. The team worked on creating UI mockups and on drafting user stories.

Iteration 1: The stories for migrating/updating the Database schema were taken up first to support related application changes in the future iterations.

Iteration 2: Major requirements including creating Multiple-Choice Questions, automating the grading and report generation, support for Mathematical equations, and setting up a password recovery system were taking up during this iteration. Furthermore, the views were also modified to reflect the schema changes performed in Iteration 1.

Iteration 3: The focus during the third iteration was to fix the previously failing acceptance tests. In addition, the logic to support "Short Answer" questions was created.

Iteration 4: The team worked on enhancing the User Interface to make it responsive. Also, the evaluation for the "Short Answer" question was made robust by introducing partial checking. RSpec tests were added to increase code coverage.

Customer Meetings

Meeting	**Description of the content of the meeting**
10/03/2018	The first customer meeting was used to understand the high priority user stories and discussion was held on how to get the project live by the end of the semester. An initial video was taken to record the basic requirements and user stories of the customer, Dr Walker. Customer listed improving the questions

	and answers interface, adding math support, adding links for the answer explanations as high priority user stories.
10/18/2018	This meeting was used to demo the customer the old application running. The customer also provided with a new user story of adding short answer type questions along with the already present Multiple choice questions.
10/25/2018	This meeting was used to demo the modified database schema to the customer made as a part of Iteration 1 in accordance with the user stories provided. The team showed the customer the design that is planned for the questions and answers interface.
11/09/2018	This meeting was used to demo the interface changes, and math equations which were added as a part of Iteration 2. The team also presented all the user stories implemented in iteration 2 and discussed about any modifications required. Customer provided a new user story for evaluating short answer questions using exact string match.
11/26/2018	This meeting was used to demo the user stories implemented as a part of Iteration 3. The team presented the case-insensitive exact string match method used to evaluate the short answer questions. Customer suggested improving the robustness of the features added until then by checking for any tests missed and also provided a new story for adding partial answer match for short answer questions.
12/03/2018	This meeting was used to demo the user stories implemented in Iteration 4 and also the final application. The customer tried all the features of the application and the team also discussed about the deployment of the application and its maintenance with the customer.

BDD/TDD process

We followed the development process which was introduced in the class itself where we wrote cucumber tests which were used as acceptance tests and RSpec tests which were used for function level testing. Since most of our features required making modifications to the existing product, many times we just had to add extra tests pertaining to our changes instead of writing the new feature/Rspec files from the start. However, many tests were missing, so during the course of development, we added a lot of new tests. We targeted about 1-2 user stories per iteration that allowed to make sure that we thoroughly test the features before deployment instead of adding tests towards the end to increase coverage.

Configuration Management Approach

To make sure that our production environment is clean, we used branch protection rules. We decided that no one should be allowed to directly push changes to “master” and hence we created rules that required developers to have their changes reviewed by at least two other developers. As a result, each developer would raise a pull request and upon being reviewed by two other developers, the changes will automatically get merged with the master branch. This process really helped us and most of the time our production environment was in a deployable state. None of the developers faced any major merge conflict issue too. We had a total of over 45 branches (some have been deleted) and 4 releases - one for each iteration.

Issues with Production releases

Heroku does allow automatic deployment where we can specify our production branch and upon every merge/commit to that branch, Heroku will run some automatic scripts and deploy our latest changes. However, this feature never worked for us. We pointed the automatic deployment to our "master" branch, which we tried to keep clean, but the automation script always failed. We did spend some time to see the issue but it appeared to come because of the ruby version that we were using. Unfortunately, one of our team members had to always deploy changes manually because of this one time we thought that our latest changes have been deployed but during the demo to our customer, the latest changes were missing. We showed our development changes during that time which was working fine. Had this automatic deployment worked, we could have avoided this manual deployment.

Issues with AWS, Cloud9 and Github

We didn't face any issues with the Development environments.

However, the production instance in Heroku doesn't provide a free SMTP service. Hence, we had to perform a workaround to support the Password Recovery feature.

Development Tools and Ruby Gems

The inline mathematical notation feature was implemented using the javascript library Mathjax. The choice of this library was made considering its LaTeX markup and its extensive documentation.

The partial check for "Short Answer" questions was done using the fuzzy-string-match Ruby gem.

We used SimpleCov to generate the coverage report of our Ruby code. SimpleCov provided a simple method for generating test coverage reports.

Final Customer Interview : https://vimeo.com/304276474
Application Demo : https://vimeo.com/305544529

Important Links

- **Pivotal tracker** https://www.pivotaltracker.com/n/projects/2200809
- **Github** https://github.com/aditya30394/Self-Evaluation-Portal-CSCE625
- **Heroku** https://shielded-caverns-97668.herokuapp.com/