Online Education Framework with Customized Problems

Jayme Green

Abstract—Online education allows for customization of education for individual users. In this report, a new online education platform was created containing a few lessons and numerous problems. Baseline testers were given the program with randomized questions and asked which questions did they prefer, what questions did they get the most of, and how many times did they skip questions. The program was then modified to customize problems in two ways: one using K-Means clustering where problems are clustered and ranked based on the users’ rating of each cluster and one which does a K-Nearest Neighbor-inspired approach to find users with similar opinions’ favorite problem. A few weeks later, the same testers received the program - but with the customized problems - and were asked the same questions. Users much preferred the customized problems approach as the average, percent difference between what questions they preferred and what they received was nearly 19% in the baseline while only 4% in the customized version. Also, on a scale of one to five (with one being never skipping problems, while five indicating always skipping problems), the baseline users skipped an average of 3.18 while the customized users skipped an average of 2.14. The customization of problems increases user experience with education.

I. INTRODUCTION

With the introduction of the Internet, numerous online education providers have emerged. These education platforms enable their students to educate themselves online; the students complete lessons at their own speed and cannot learn new material without a level of comprehension from the current material. While this system allows for more customization than normal class-like settings, it remains linear. The questions generated and given to each student is the same. The difficulty of the material remains the same. And the presentation of the material remains the same. Students are not the same.

Students have different motivators when it comes to learning. Some want mastery of the subject, some want working-knowledge of it, and others just want to pass their class. Why have a single, linear education path when students do not have single, linear motivations? To resolve this problem, this project will introduce an application that will customize learning for the user focusing on problem presentation.

The project has been broken into two main parts: the creation of the online, education platform and the algorithms used to customize the education experiences. The discussion of the educational platform lies in the Design section which discusses the creation and capabilities of the platform as well as discusses the design considerations. This program - a Microsoft Visual Studio C# program utilizing a SQL database - drives the capabilities of the algorithms derived and described in the Solution section. For the customization I utilized K-Means to cluster problems to give the user their most preferred problem as well as giving problems that other users with similar opinion liked using a K-Nearest Neighbor-inspired classification.

In order to evaluate the performance of the customization, 22 users utilized the application when the customization algorithms were used as well as when the customization were not enabled and given problem. These users gave feedback on which they preferred and majority received question types that they preferred with them skipping problems less often than the baseline tests. The testing and results are further explored in their respective sections.

II. RELATED WORK

Online education has been growing rapidly; numerous conferences and workshops have emerged focusing on the advancement of online education [1]. The biggest advantage emerges with customization. An online program can analyze how well a student understands each problem. It can predict when a student might get stuck on a problem or concept [2]. Unlike traditional classrooms, the tool can give the student an additional lesson or further support. Also, it can interpret whether a student’s mistake resulted from a true misunderstanding or from a small typo [3]. These minor mistakes will happen and need to be categorized properly. A student should progress as they understand the material at their own speed.

Additionally, students learn differently. While textbook reading might help one student understand a topic, another student might thrive with discussion-based learning [2]. By learning and predicting how a student learns best, online education can teach millions of students with individual, customized lessons bringing significant benefits that a traditional classroom struggle to rival.

Many of the challenges of online education emerges from customization. How do you evaluate how well a student understands a concept? Most proposed answers utilize machine learning techniques [1]. Linear regression, random forest, and k-NN have been applied to predict whether a student will get a given question right or wrong [4]. These functions had ranging success; the amount and quality of data influenced the results greatly. Other techniques, such as using boolean logic analysis, emphasize variables like
question difficultly [3]. While all of these techniques have found success, they have yet to be experimented in large platforms.

### III. Tools

The main development tools used for the project were Microsoft Visual Studio and Amazon Relational Database Service (RDS). Microsoft Visual Studio offers a lot of versatile tools that help with development. The biggest strength revolves around the .NET framework offered. The .NET framework provides all of the front-end elements. As this project was mostly focused on design, back-end development, and customization of problems, the front-end was secondary and easily handled by Visual Studio. Additionally, Visual Studio and the .NET framework offers easy integration of numerous databases. As the program requires a database to store the problems, lessons, user data, and more, the ease of integration as well as variety of tools with the database was also a major advantage. This made all development able to be in the same program (Visual Studio) with more than enough tools to help development.

As Microsoft Visual Studio offered a lot of useful tools, the programming language was decided afterward. The program supports Visual C# and Visual Basic, which led to C# to be chosen due to prior experience.

The database needed to be hosted online to allow multiple users to access it at the same time from anywhere. This requirement led to only a few choices. Amazon Web Services (AWS) offers many software tools including Amazon Relational Database (RDS). Amazon RDS allows for numerous types of relational database to be made with paid scaling depending on usage; this scaling also allows for a free-tier to students with $100 credit in case the usage goes over the free-tier limit. Having the database free, online, and able to to select from different types of relational databases, these strengths led to Amazon RDS to be chosen.

Amazon RDS allows for numerous different relational databases including the three major types: mySQL, PostgreSQL, and Microsoft SQL Server. As Microsoft Visual Studio was already chosen to be the IDE platform, Microsoft SQL Server was chosen to provide easy integration between the database and software as both are Microsoft products. All of these tools allowed for development, testing, and distribution to be done without many errors.

### IV. Design

#### A. Database Design

As the program heavily depends on saving lessons and problems as well as reacting to user input, the database design becomes integral. The overall design can be seen in Figure 1. The three main ideas of the project revolve in: users, problems, and lessons. Users need to have usernames (the primary key) and passwords in order to log into the system. Once a user is added to the program, the connection table between user and lesson becomes populated with all combinations of that user and all lessons with all being labeled as incomplete. Similarly, when a lesson becomes created, this lesson-user combination is added to the connection table with the lesson being incomplete for each user.

Lessons contains an ID, title, content, pre-requisite, media, media description, and media title. As lessons can have numerous images, the fields of media, media description, and media title was placed in their own table with the lesson ID that it associates with. While the lesson id is used as the primary key in the database, the user sees the lesson title as this is much more descriptive than the ID. The only field that can be null in the table remains pre-requisites (which can contain another lesson ID) as a lesson can be open to any user immediately.

Lessons need problems in order to properly work. Therefore, a lesson problem connection table was created that specifies what problem associates with what lesson using their IDs. Similarly to the lesson table, problem’s primary key was an integer id. Problems also had a title, content, difficulty, average user rating, and choices. Average user rating specifies the average user rating given by all users who completed this problem between one and five. This rating is used for customization of the problem clusters as it indicates overall, favorable problems.

The choices, as there will be numerous, were put into their separate table containing the choice, feedback, and whether it was the solution or not. For dropdown and multiple choice questions, the amount of choices for each problem ranged from two to six, while the chart had only two choices: a given configuration and the answer configuration. This was due to the user having unlimited amount of variations; as a consequence, users get less feedback on charts.

Lastly, as users attempt problems they generate history. Therefore, all of their responses are saved in a history table connected to the user. History contains what user, lesson, and problem it originated from as well as what the user specified, are they correct, as well as their rating for that problem. Therefore, this table can see what individual users thought of a problem as well as the number of attempts they did on it. This table became integral for the customization algorithms described in the Solution section.

#### B. Framework Design

The education framework was completed in Microsoft Visual Studio using the .NET framework and C#. The .NET framework allowed for easy front-end design with event-handling communication taken care of. This integration
allowed for the model and database communication as the
most integral parts to handle in the program. The overall
program flow of the education framework can be seen in
Figure 2. The six main pages are:

1) Log In: The first page that any user sees when starting
the program is the log in page. As seen in Figure 3, this
page prompts the user to provide a username and password.
If the user does not have an account - or wants to create
a new one - a register button also resides on the bottom
of the page. Once clicked, the register button opens a new
window for the user their preferred username, password,
and email. The username can be anything the user prefers
as long as another user has not claimed it first. As the
program compares usernames by converting all characters
to uppercase, the username that the user selects must be
unique in at least one character. The password that the user
picks must also be given twice: one for a password field
and another for confirm password. These two passwords
are compared with each other then encrypted as it enters
the database. Encrypted with SHA-256, the passwords are
protected as SHA-256 encryption has yet to be broken. The
email field is optional for the new user; as the user can
put whatever username they want, an email would be an
easy identifier. Once registered, the user becomes added to
the database and identified as non-admin. In the platform,
only one admin was created manually. This manual creation
allows it to be impossible for a normal user - without the
source code - to create an admin user.

2) User’s Main Menu: Once the user logs in successfully
into the platform, the database checks if the specified
user is an admin or not. If not an admin, the user’s main
menu opens as seen in Figure 4. This page contains a
welcome as well as dropdown indicating what lesson(s)
they can attempt. Lessons are only shown when the user
has completed its’ pre-requisites. Hence, a new user will
see only lessons with no pre-requisites. Upon selecting a
lesson, the user can press a button to launch the lesson
and connected problems. This dropdown updates when the
user completes a lesson, including all lessons that have a
pre-requisite of the recently-completed lesson.

3) Lesson/Problem Page: Once a user - admin or normal
user - selects a lesson, the lesson page opens and populates.
As seen in Figure 5, the left side of the page has the
lesson including the title, content, media title, media (as
a .png, .jpg, and .gif), as well as media description. Only
one image is shown at a time. The user can go to another
image by using “Next” and “Previous” buttons which

Fig. 5: The page where users can see lessons and complete problems.

update the media, title, and description. On the right side of the platform, the problems are displayed. The problem includes the problem title, the question, and the choice(s) which depends on the problem type - either multiple choice, dropdown, or chart.

On the bottom of the page holds the a user rating score, submit button, and a skip problem button. Before the user can successfully submit their answer they must rate the problem. This rating must be between one and five (with five the highest) which indicates how much a user liked this problem. While the users are required for every submission, the learning algorithms (as explained in Solutions) only uses the ratings of when they get the problem correct. Regardless of whether the user gets the problem correct or incorrect, their response is stored in the History table in the database storing the problem, lesson, their response, whether they correct, and their rating. The user also has the option of skipping a problem to go to another problem in the lesson. As problems have different type and difficulty, skipping allows users to skip problems that they dislike.

In order to complete a lesson, the user needs to get 15 problem points. These points are received by completing problems with each problem giving a different amount. Difficulty of the problem, specified during problem creation, determines the amount of points. The difficulty ranges from one and five resulting in the minimum amount of problems required to pass to be three and the maximum to be fifteen. This allows users who prefer repetition to get a lot of problems while others who want to progress fast to do so as well. Points are shown when the user gets a question correct. Once the user gets at least 15 points, they will be prompted of the completion, have their lesson close, and have the dropdown of the lessons they can do updated.

4) Administrator's Main Menu: Once an administrator logs in, they are brought to the admin’s main menu page seen in Figure 6. This page simply has buttons that allows the admin to access pages to add a lesson, add a problem, and view (and do problems of) a lesson. Due to just being used and seen by the administrator, the buttons are placed in front with little formatting.

5) Add New Lesson: The add new lesson page allows the admin to add lessons to the platform as seen in Figure 7. This page has fields for lesson title, content, pre-requisite, media, media title, and media description. There are also a few fields like audio rating and visual rating that are part of the form and stored in the database but not actually used in the program. This was due to having this page being one of the first developed pages. Ambitiously, the project could have done customization of lessons as well as problems. Therefore, these fields were included as it is much easier to include, add to the database, and ignore than to not include these fields, do not add to the database, and discover that I needed them.

The program utilizes all other fields. The lesson title and content accept strings where the title must be unique. The media can be any .jpg, .png, and .gif from the admin’s computer; the administrator can add as many images as they want. With each media, the user can put a title and description to help describe and reference the image. Unfortunately, the media must be added as part of the lesson. If the administrator wants the users to reference an image in a problem, the admin must add it in the lesson. With limited time of development, the admin cannot edit a lesson making the administrator have to plan their lessons and problems prior.

Lastly, the administrator must specify if this lesson relies on another lesson as a pre-requisite. The pre-requisite field, a dropdown, holds all lessons in the database; the administrator can either select a lesson or leave the field blank - indicating the lesson can be accessed by any user immediately and does not rely on another lesson’s completion. While this process works, there are two notes that the administrator has to know: the pre-requisite lesson must be created before the current lesson as well as the
lesson can only have a single pre-requisite and cannot have co-requisites. Therefore, the administrator must make the lessons in order of which they want the students to do them in (assuming they are all on the pre-requisite chain). Also, although each lesson can only have one pre-requisite, a lesson that has a pre-requisite can be pre-requisite for another lesson. Additionally, a single lesson can be a pre-requisite for multiple lessons meaning that if a user completes that lesson, numerous lessons can be unlocked.

6) Add Problem: The administrator can also add problems to any lesson in the database as seen with Figure 8. The user needs to specify the problem’s title, the problem prompt, difficulty, and what lesson it is associated with; all of these fields are required for every problem. The admin then has to specify what type of problem they want to give. Depending on their specified check, new input fields are brought up.

For both multiple choice and dropdown, six pairs of choice field, is correct checkbox, and feedback field pops up. The administrator has a choice of including two to six choices for the problem. Choices need to have both the choice and feedback field filled out with the program giving an error and not adding to the database if only one is filled. Also, at least one of the choices has to be checked to be correct. This allows the administrator to have multiple choice and dropdown to have multiple correct answers. For the dropdown, the users just need to select one of the correct answers while the multiple choice requires the user to select all of the correct answers to be correct.

If the chart has been select, two charts appear to the administrator called: given and answer. The given chart represents what should be presented to the user while the answer chart represents the answer that the administrator wants. The chart becomes allows the user to have eight columns with an unlimited amount of rows. As the chart's information converts into a comma-delimited string before being stored in the database, the administrator can choose to have the given and answers with different dimensions. Due to being stored as a string, the user must match their answer with the solution answer exactly to be right. This requires the administrator to create clear instructions and given chart to the users.

V. Solution
A. Customization via Problem Clustering

The main idea of problem customization revolves around trying to give problems that the user likes to the user. Therefore, the problems that are most similar can be grouped into a cluster. As the user does problems and rates them, their opinions of a cluster will be updating. As the clusters will contain similar problems, problems that they did not complete from the same cluster of their favorite cluster will also, most likely, be liked. This idea of clustering problems and giving their problems from their favorite cluster has been implemented by using K-Means and sorting the results.

K-Means is a unsupervised algorithm that clusters points together into k clusters. Firstly, the algorithm randomly places the k-cluster-centers within the dimensions of the data. Secondly, each data point is assigned to the closes cluster-center using euclidean distance. These cluster-centers then move to the center of mass of their assigned data points. This process repeats with the data points assigning themselves to the closes cluster-center and then the cluster-center moving to the center of mass until the clusters become stable.

K-Means clusters problems fairly well but relies on three main decisions: how many clusters to use (k), what data to cluster on, and how many iterations of K-Means do you do to find the best results. The amount of clusters can be determine by experimenting with a few different k values. This allows numerous k values to be tested, the best to be found, and without human intervention. In this project’s
implementation, k values are experimented starting from two (and one cluster would not be useful) and continuing until the improvement of the sum of squared error lowers to below 30%. Sum of squared error measures all of the error that the clusters have. As each point, ideally, wants the center of the cluster to be on itself, the error emerges in the euclidean distance between the actual center and the data point. Also, as the clusters are determined by a random center, there are chances where this random skews the results. Therefore, each k value is repeated twenty-five times with the iteration with the best sum of squared error is kept to compare with others. While this process requires some computation to do, it does guarantee that a good k value - and the number of clusters - is selected.

Another decision comes with what data to cluster on. As I want to cluster problems, I need to use data that will best separate the problems. Therefore, I chose to use difficulty and type properties. The difficulty indicates how difficult a problem is while type represent what format the question is in - either in multiple choice, drop-down, and chart. These two attributes were chosen as some users will like certain type of problems (I, personally, like multiple choice the best) as well as some users like different difficulties. As the lessons need fifteen "difficulty points" to pass, some users may prefer to go through lessons fast with difficult problems while others may want to slowly go through it. As these attributes seem to be the most important, I did not want to put other, insignificant attributes.

The last decision made was how many K-Mean iterations to do remains important due to the random element of starting the clusters. Some of these starts will better than others leading to lower sum of squared errors overall. With this knowledge, the more iterations done of K-Means, the more likely the best cluster will be found; yet, more iterations requires more time. Therefore, I chose to do fifty iterations as this does not take too long and the best clusters in fifty iterations will, most likely, be a good enough clustering.

With the problems clustered, each cluster is ranked based on the rating that the user gave problems that they already did. Then, each problem in the cluster (that is in the lesson of that the user is taking) that has not been taken is sorted based on average user rating. This ordering represents the problems that the user will like from most likely to least likely.

B. Customization via User Opinion Selection

Another method of customization tries to give the user problems based on other user’s opinions. As each user gives a user rating for each problem, the current user can see which other user agrees with them the most. With similar opinions, the user can be given the highest rated problem that the comparable user has done (that the user has not).

With every problem done by the user, the closes opinion use could change, but this method could help maximize the user’s rating.

To find the best problem using this technique, the most agreeable user must be found. This was done by iterating through every problem the user has done and comparing their rating and number of attempts to every other user. The difference between the rating and number of attempts for the user and other user is measured using euclidean distance. Going through each problem that the user has taken, the average agreement rating is discovered by taking the total distance calculated divided by the amount of problems that the users shared. As some users would have shared no problems with the users, they are given the middle value of 2.5.

All of the users are then sorted based on this agreement factor with the most agreeable user first and the least agreeable user last. As new users are being registered, the amount of users to consider as most agreeable will scale with the amount of total users. With more users considered, the consistency of the end user getting a problem they like increases. Therefore, the amount of user considers becomes calculated by taking the top 10% most agreeable users (with 1 user always being considered if there are less than ten users in the system).

With these users selected, the questions that they took (that exist in this lesson and that the user has not taken yet) are ranked based on their user rating. This ranking represents the ordering in which the users with similar opinion think are the best problems. With similar opinions, the user will also, most likely, rank this problem highly as well.

C. Combination of the Two Customization Rankings

With the two rankings provided by clustering problems and by ranking based on similarity of user opinion, these two methods could provide different rankings. Therefore, these two rankings are merged. The problems that are ranked first are given a score of one, the second ranked given a score of two, and continued. These problems are finally sorted with the lowest score problems first with the highest scoring problem last. The problem in first becomes the problem that is given to the user.

The problems are ranked due to users being able to skip problems. If a user does not like the problem, they can skip to the next one. The ranking makes this process seamless as the next problem that the user is predicted to like would be in the second position and the problem they would probably hate as the last problem requiring numerous skips. This process of ranking problems in two ways will lead to the users being happier - giving higher ratings - than the
randomized problem that was given in the baseline testing.

VI. TESTING

The program was converted to an executable and given to about 30 friends and family to complete. I worked with another RIT student, Chance Beagley, who created the lessons and problems for his senior project in the School of Individualized Studies (SOIS). Three lessons were created that chained together as pre-requisites: binary math, logical gates, and history of measurements. Each lesson had about 15 problems each with ranging difficulties and types. While the number of lessons and problems were relatively low, this allowed the testers to spend more time trying the program and give proper feedback. A new user can complete all three lessons in approximately 25 minutes.

These users were given one week to complete the lessons with their attempts and progress tracked by the database. Additionally, the users were asked to fill out a short, Google Forms survey about their experience. For both testings, the survey included questions revolving what question type did they prefer, what question type did they get the most, and how many times did they skip questions.

The only differences between the baseline and experimental testing was how the program decides how to give a user a problem. The baseline gives the user a random problem from the lesson, while the experimental test uses the customization described in Solution section.

VII. RESULTS

A. Baseline

Out of the 30 friends and family that were given the program executable, 22 of them went through the program and filled out the Google Form survey. Two of the questions offered in the Google Form survey included: what question type(s) did you prefer as well as what question type did you receive the most. These two questions were combined into Figure 9.

Ideally, users should get problem types that they prefer. But with the baseline, the user gets a random question. Therefore, as reflected in Figure 9, the users preference and what they got are very different. Ideally, these bars should be equivalent to each other as users who prefer charts should get chart questions the majority of the time. Unfortunately, these bars cannot be equivalent due to how the questions were asked on the Google Form survey. In the survey, the users were allowed to pick one or more options for which type of problem they prefer, while they only could pick one option for which problem did they get the most of. To reflect this difference, percentages were put onto each bar to present how many replied with that response comparatively to the total amount of votes. Therefore, the goal of the experimental results should show the pair of bars evening out to each other and having their percentages the same. For the baseline, the average difference between the bars was 19%.

Another metric for evaluation was asking how many times did the user use the skip question button. This button allows the user to skip to another question in the lesson. By utilizing the skip question button, the user indicates that they did not like the problem that was presented to them. Therefore, if the questions are presented questions that they enjoyed, they will skip questions less. The users of the baseline were asked how many times they used the skip question button on a scale of one to five; one represents that the user did not know the skip question button existed and five indicates that they used this button to surf through all questions to pick their most preferred question. The results of the survey can be seen in Figure 10.

As seen in Figure 10, the users skipped problems often averaging 3.18. These results do make sense as the users were presented random questions. Ideally, with the customization of problems, the users would like the problem they were presented and not skip as much. The experimental results show the improvement where users do not skip
The graph shows the preference of question type among the experimental users compared to the questions the application gave them according to a Google Forms survey. Note: the color bars do not sum to the same result. This was due to the preference survey questions allowing multiple questions to be selected, while the question asking what the users received the most only allowed for one selection. The percentage on the bars represent this skew.

as much implying that they received questions that they preferred.

B. Experimental

After implementing the two customization algorithms as specified under the Solution section, the same users that participated in the baseline testing were given the program with another Google Forms survey. The results received for the experimental showed the product improved noticeably from the baseline.

One of the main metrics was comparing whether the users received problems that they preferred. As seen in Figure 11, the users were asked what questions did they prefer and what questions did they get the most. Ideally, these bars and percentages would be even. Yet, while these bars not even, they are close with their percentages (the real comparison) being 5% at worst. These results are incredibly promising as compared to the same question asked in the baseline in Figure 9 where the best pairing was 13% - with the worst being 29%. The average difference in the baseline was 19% while the experiment had just a 4% difference. These results prove that the customization drastically improved which problems were presented to the user giving them problem types that they preferred.

The other metric used for experiments was measuring how many times the user utilized the skip question button. By using the skip question button, the user indicates that they did not like the problem presented to them and prefer another one. Therefore, if the user likes the presented question, they will skip less. The results are shown in Figure 12.

The users reported that they used the skip button occasionally to a few times with a decent amount never using it. Comparatively to the baseline results in Figure 10, the users skipped problems less. For the baseline, users gave an average of 3.18 while the experimental had an average of 2.14.

Yet, a lot of users still skipped problems. Two possible reasons why the majority of users ranked it two and three could be: the users have different definitions of what each number represents as well as the program giving a few random problems to get more data from the user. Firstly, the users were given this survey question with only the extremes (one and five) defined. Therefore, every user has their own definition of what the middle values mean. This causes the difference between the baseline and experimental to be more significant at the extremes: in the experimental, significant amount of users never skipped questions compared to the baseline while zero users skinned through questions in the experimental. Also, the customized version requires data from the user and gives the user two random questions in the beginning to answer before trying to customized. Additionally, every new lesson gives another random question to prevent overfitting. These random questions are more likely to be skipped as no customization has been done. While these results are not as substantial as the previous metric, they show that the customization algorithms provided a better user experience than without.

VIII. Conclusion

Education remains an integral foundation in every person’s life. And the way a person receives this education can change how much they retain as well as their attitude to receiving more. With the rise of Internet, online education can provide an avenue to customize the way education is presented to the individual.

In this project, I created a full .NET, C# program with a Microsoft SQL database that contains lessons which users can answer questions. With this program, users were given two versions: one with randomized problems and one with problems given based on problem clustering as well as similarity of opinion with other users. The customized version provided a much better user experience as users
received problems that they preferred as well as did not skip problems as much.

While this program has not solved online education, it does propose and produce successful methods to improve it.

IX. FUTURE WORK

Although the results are promising, the testing sample size was only 22 users who were friends and family. With a low sample size, the results cannot be fully confirmed. Additionally, as the group was friends and family, they might have been biased in their responses. While I reminded them numerous times that their responses were anonymous and to be honest (even if their responses were incredibly negative), the relationship might have caused more positive responses. With a larger test group of random individuals, the results can be confirmed definitively.

Another aspect of the program that suffers emerges in the amount of computation required for the customizing algorithms. As the K-Means clustering of problems requires numerous iterations as well as the opinion requires lookup of every users problem history, this requires a decent amount of computation. The users reported that it took nearly two seconds for the program to give them a problem when they opened a lesson as well as when they got a problem right. As the time of this project was approaching, my implementation of both algorithms was quick where results were more integral than the efficiency. This, most likely, caused non-optimal implementation which can be improved with a code review.

REFERENCES