Final Report: Parsons Coaching System

Da An da2841, Yuan Tian yt2583

May 2019

Contents

List of Figures

1 Abstract

This report summarizes our work to the Social Addictive Gameful Engineering (SAGE)[4] project. Our main focus is on the implementing of Parson's coaching features under Gameful Direct Instruction module. We implemented a hierarchical feedback system enabling the instructor to provide different kinds of feedback to students. Feedback can be an incentive to encourage students to continue improving their skills and learn CT Concepts in the right direction. We aim to design a system that is not only highly customizable but also highly efficient. We also implement a labeling system to enable instructors to give each game, quest and mission labels about their focusing areas. Moreover, we include several user interface changes on the student side and instructor side, including the dialogues between avatar and students, the pop-ups for choosing/writing feedback and CT Concepts.

2 Introduction

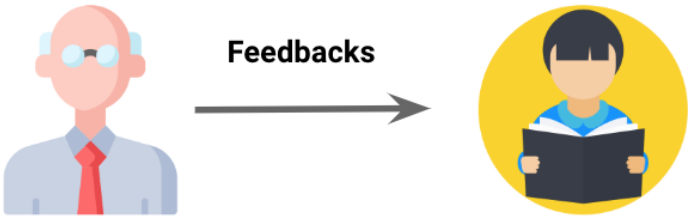

Figure 1: Interactive Feedback from teachers

The current Social Addictive Gameful Engineering (SAGE) project aims to provide students in grade 6-8 curricula with enjoyable, game-like learning experience as well as enhance their programming skills and improve their Computational Thinking Concepts (CT Concepts) in the meantime. At the beginning of this semester, a gameful direct instruction system has not been fully implemented. Teachers could not give students customize feedback and students could not interact with teachers based on the feedback from teachers for each move and general assignment performance. Also, the two-level CT Concepts have not been able to persist in quest and game level.

This semester, we focused on build a feedback authoring system. A system that, on the instructor side, designed to give maximum freedom to customize the feedback and also achieve efficiency, and on the student side, designed to give students a more interactive environment. We call the system the "hierarchical feedback system". In the current SAGE design structure, there is a hierarchical system build on the games: games comprise quests, quests comprise missions. The feedback system is sitting on this system. However, we have to make some

modifications to this because the relationship among each level is not merely one-to-many relationships. We then divided the feedback into two categories: move feedback and assignment feedback. Move feedback is the kind of feedback that shows up when a student makes a single move in the sage-scratch panel. The move-feedback is further divided into four classes: correct, incorrect, neutral and close. When a student makes a move during an assignment, the instructor can give move feedback according to this action. In this way, students can get instant feedback on each move. Assignment feedback is the feedback that shows up after students make a final submission. Instruction can input 3 classes of feedback (proficient, developing and basic) before students take the assessment. Then sage-frontend will show the feedback based on the student performance.

We also build a labeling system for quests and assignments. The labels come from a hierarchical system of the computational system. Each game/quest will be associated with several computational thinking concepts (CT concepts). Instructors can use these labels to provide further customized feedback, manage assignments, etc.

On the student side, we mainly work on the student assessment interface to incorporate the newly designed move feedback and assignment feedback. We also centralize the conversation between students and front-end during an assessment by moving all the messages (such as hint, instruction question and move feedback) into a scroll-able dialog box under the avatar.

Our work mainly stays within the Affinity Space and Sage-login database. In this report, we are going to discuss in detail how each module works from data model design to user interface design, and finally to code structure. We are also going to talk about how do those modules going to correlate with others. After that, we are going to shows the list of bugs that we fix. Finally, future work is included so that people after us can get a better understanding of where we are heading.

3 Backgrounds and Related Work

3.1 Backgrounds

Scratch, aims at teaching advanced computer programming concepts, has become a popular tool since it was published in 2001 from MIT. Scratch allows users to program interactive stories, games, and animations using a simple drag and drop interface[1]. However, there is little guidance in the scratch side to offer students customizable feedback and general summary of students performance after the completion of each game. We sought to improve the user interface of student assessment panel with scratch game embedded in and offer students timely move feedback on each move and performance feedback after they submit the solution.

Why is feedback so important in coaching? We believe that timely move feedback can be an incentive for students to continue games besides hints, and general assignment feedback can help students find their strengths and weak-

nesses so that they will be able to focus on training on certain games with certain CT Concepts. This helps to achieve the "Addictive" goal in the sage project.

"Scratch – a programming environment that enables young people to create their own interactive stories, games, and simulations, and then share those creations in an online community with other young programmers from around the world." [6]. CT Concepts based on the context of sage scratch plays an important role in guiding game design. This is because CT Concepts are common in many programming languages and train CT Concepts for students in grade 6-8 curricula can help them lay foundation for future's logical and systematic language study in computer science field, as students are able to apply the same concepts to break down big problems to small sub-problems and test solutions using recursion, iteration, and sequence testing methods. However, as is said in previous papers, there has not been a uniform definition for CT Concepts."Although computational thinking has received considerable attention over the past several years, there is little agreement on what a definition for computational thinking might encompass (Allan et al., 2010; Barr & Stephenson, 2011; National Academies of Science, 2010)."[6]. This semester, we extended the CT Concepts and linked them to each scratch game and quest based on previous papers.

3.2 Related Work

We refer to related work in two categories. First, we sought to find some related work showing the similarities to the existing sage platform which can provide automated feedback and coaching for students. Second, we refer to some previous work on the classification of CT Concepts and classify CT Concepts into three levels and later linked the CT Concepts to each game.

3.2.1 Parson's Programming Puzzles

Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses (Parsons & Haden, 2006)

In this paper, the design of puzzles was introduced to ask students to place code lines in the correct order with each code lines provided in advance. Although students can get feedback like the percentage of right items, they are not able to get the reasons for wrong item placement. The authors also pointed out that more stylish user interface could intrigue students so that the game could be facilitated. We provided students feedback move feedback and the feedback could be constructed to provide students with reasons for the wrong move. Also, we added Emojis to both instructor panel and the students' assessment game panel based on students' game performance to add fun as our students are mainly from grade 8 to 10, and we believe this will help intrigue students and bring pleasure while students are playing games.

3.2.2 Learner Problem Solving Strategies

Online Identification of Learner Problem Solving Strategies Using Pattern Recognition Methods

This paper introduces feedback given by system messages based on procedural knowledge of the learners in computer science education as well as hidden Markov Chain with pattern recognition methods. The system will automatically generate the learning strategies. Our feedback system, however, rather than generates learning strategies using machine learning method, generate assignment feedback from instructors, which could be used as future learning strategies.

3.2.3 Case-Based Reasoning

Using Case-Based Reasoning to Improve the Quality of Feedback Provided by Automated Assessment Systems for Programming Exercises.

This paper introduces some factors that can affect the students' learning outcome. To find the similarities among computing incorrectness, the author explores static analysis of source code, execution traces of running programs and compares outputs among test cases.

3.2.4 New Frameworks for Studying and Assessing the Development of Computational Thinking

This paper identifies seven CT Concepts in the context of scratch and these concepts could be used in other languages. The seven CT Concepts include sequences, loops, parallelism, events, conditionals, operators, and data. The author then gives a detailed definition of the seven concepts. Our project extended the current CT Concepts used in sage project based on this paper then classified the CT Concepts into three levels: the primary level, the second level, and detailed level and later linked these concepts to each quest and game.

3.2.5 CSTA K-12 Computer Science Standards: Mapped to Common Core State Standards

"This document provides comprehensive standards for K–12 computer science education designed to strengthen computer science fluency and competency throughout primary and secondary" schools. [2] We extracted some standards which include: problem statement, problem exploration, parallelization, algorithm definition, data representation, visual representations for each game models and simulations accuracy, sequence, selection, iteration, and recursion. There are some overlapping with the previous papers and we adopted the overlapping concepts.

3.2.6 Creative Computing: Learner Workbook

This book introduces the scratch game by module and what will be cover in each module. We refer to this book to find the focus for focuses on scratch

games.

3.2.7 Creative Computing: a design-based introduction to computational thinking

This book offers a guide for those who have to need to design games to "support students' development of computational thinking through explorations with Scratch"[3]. We refers to the part"Computational Thinking Connections"[**?**] in this book. We chose the concepts mentioned in this book and these concepts are almost consistent with the CT Concepts mentioned in 3.2.4.

3.2.8 Creative Computing[5]

This book gives detailed guidance on Scratch. We looked into the LEARNING OBJECTIVE and KEY WORDS, CONCEPTS& PRACTICES part in each module to summarize the CT Concepts. The CT Concepts we found are basically in accordance with the CT Concepts we found in previous papers.

4 Architecture

At the beginning of this semester, the student assessment page of sage project does not show hints under Avatar. We have moved all Parson's communication controlling interface to the Avatar. Also, we had a controlling panel for the Avatar. Students can make submissions, ask for hints through this controlling panel. The dashboard will periodically retrieve the data of feedback by polling the assessment Server. The data will then be shown under avatar and pop up window.

The diagram 2 shows the interaction between teachers and students via sage log-in server and the web server. At teachers' side, the teachers will write assignment/move feedback for each quest and each game with the students' name and save the feedback in sage log-in database. At the students' side, they can retrieve the data of feedback from avatar and get general assignment feedback after their submission.

To increase efficiency and save instructors' effort when they are writing feedback for each quest and game, we come up with a design to give instructor maximum freedom to customize the feedback by inheritance from upper-level feedback and on the student side, this design can also provide students a more interactive environment. The diagram 3 shows the logic for feedback inheritance from quest level to game level. When teachers are writing feedback, The top level is quest feedback. As multiple missions can have the same quest, we limit the access and inheritance from the mission level to the quest level, because we would not like to see the mission level data be changed by different quests. As for quest and game level relationship, we enable teachers to write/update game level assignment/move feedback and select CT Concepts for focus by inheritance. If teachers add/delete new game level feedback, the feedback at a game level will use the input data as feedback for game level and the data inherited

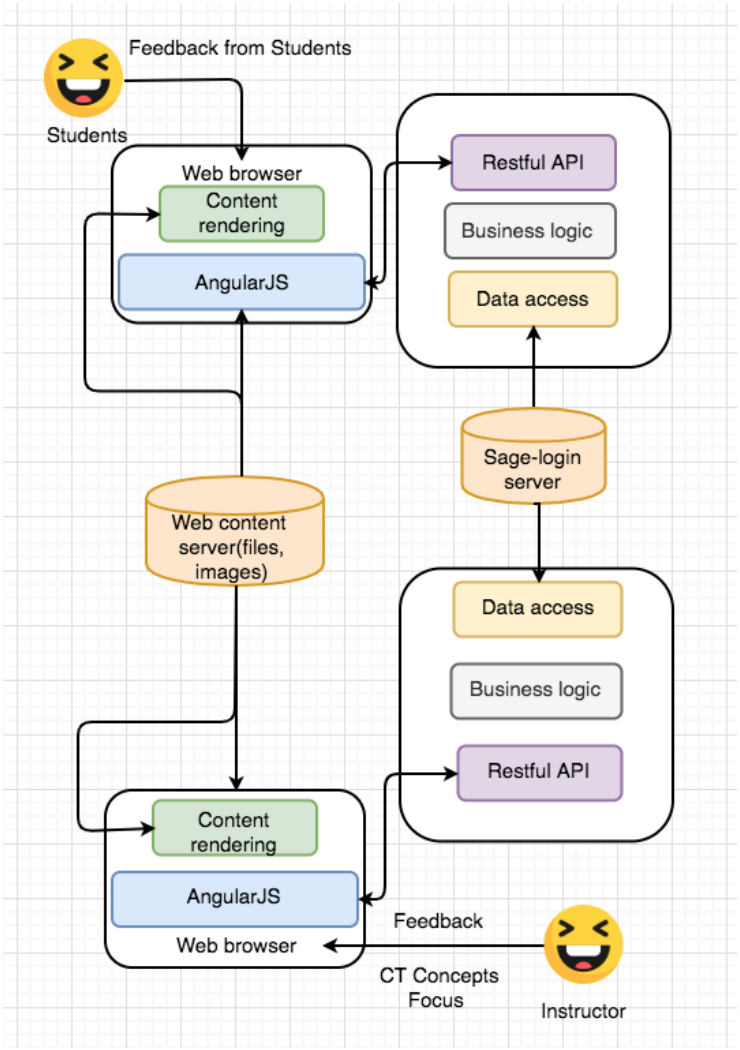

Figure 2: Data Flow for Feedback and CT Concepts

from the quest level will be overridden. If they do not put new feedback for game level, then they will be able to use the default feedback inherited from the quest level as the game level feedback. As for the focus of CT Concepts in the Computation Thinking Concepts Labeling System(CTCLS), the logic works in a similar way. The instructor will be able to refer to the upper quest level CT Concepts as focusing areas when they select game level CT Concepts. However, instructors are able to select the same level of CT concepts or different CT concepts.

For the instructor feedback authoring system (IFAS), feedback falls in two

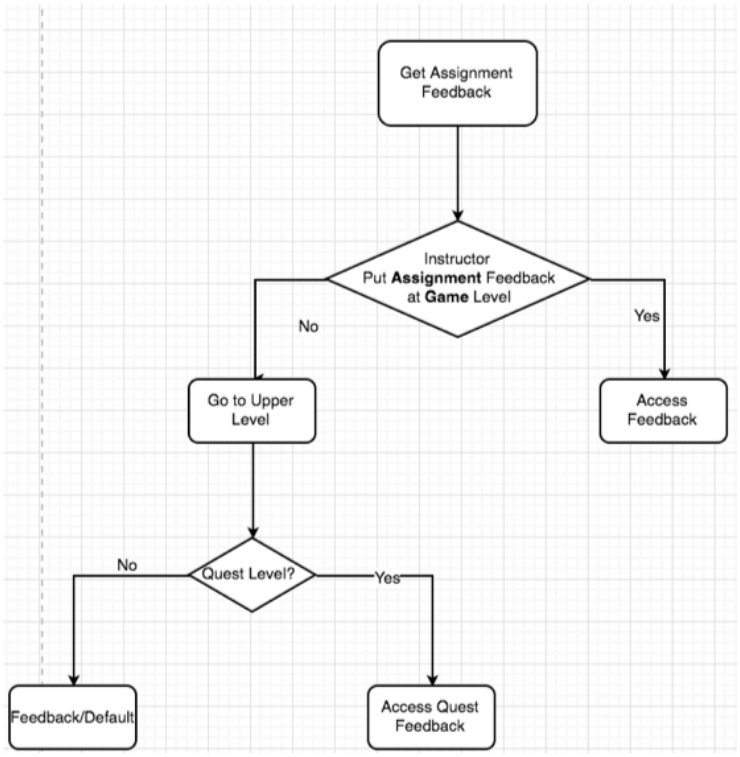

Figure 3: Hierarchical Structure

categories: the assignment feedback and move feedback. The assignment feedback is for general performance after students' final submission for each game. Move feedback is for each move in the game. The assignment feedback is classified as basic performance, developing performance and proficient performance. The move feedback is classified as four types: the correct response, the incorrect response, the neutral response, and the close response. The correct response is that the move is in the solution. The incorrect response is the move is not in the solution and is attached to the correct block. The neutral response is the move is in the solution but not attached to any blocks. And the close solution is the move is in the solution and is attached to blocks on neither sides.

In CTCLS, the focus is extended based on the papers mentioned in the previous papers and textbook. the CT Concepts is classified to three levels: the primary level, the second level, and third level and is shown in the browser in a tree structure as is shown in figure 5. The primary levels of focus provide general directions for the game including Sequences, Conditionals, Loops, Parallelism, Event, Operators and Data. The second level includes sub-categories of primary main focus like Problem Statements, Recipes, Instructions and Algorithms under Sequence, Parallelism Forever and Conditional Parallelism under

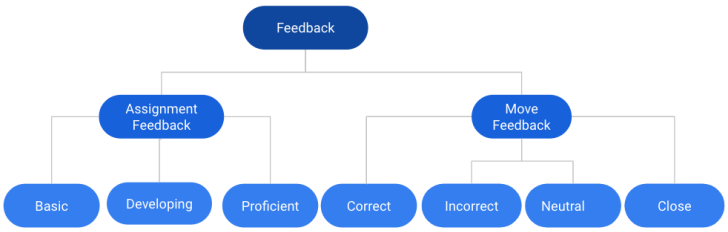

Figure 4: Feedback Types

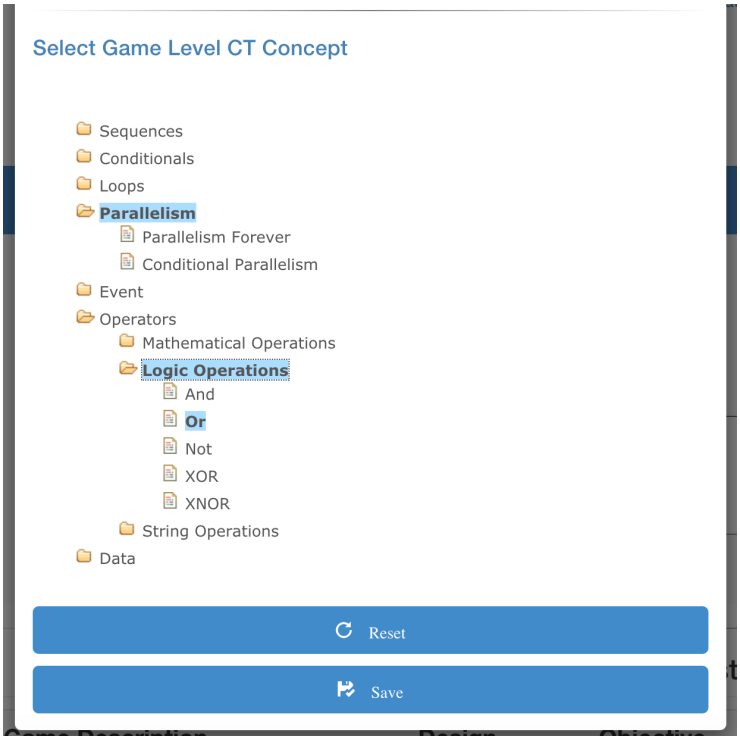

Figure 5: CT Concepts Types

Parallelism, Mathematical Operations, Logic Operations and String Operations under Operations and so on. The detailed level includes more detailed CT Concepts like And/Or/Not/XOR/XNOR operations under the second level logic operations.

5 Implementations

5.1 Instructor Feedback Authoring System

To boost the interaction between students and teachers and enable students to do active learning by learning from hints and feedback with a special focus. We designed an "Instructor Feedback Authoring System"(IFAS). This system enables teachers to create/update move/assignment feedback for quests and games. At the game level, teachers will be able to use feedback from the quest level by inheritance or override the quest level feedback by creating game level feedback. This design gives teachers maximum freedom to customize the feedback and will allow them to use to quest level feedback at any time in the game level without having to explicitly edit each game level feedback.

5.1.1 Features

Inherit quest level assignment/move feedback for each game At the game level, Teachers will then decide whether or not to adopt the feedback. If teachers put the feedback for a specific game, then the quest level feedback will not appear on the student side.

Add up to Five Comments for Feedback Each Time When creating feedback at the game and quest level, teachers will be able to add multiple comments of feedback at the same and those short-history inputs will show in the area below the input box. They can then send them all together and the new feedback will immediately show up in the right panel.

Delete feedback Teachers will be able to delete assignment/move feedback in the right panel of the updating pop-up window. They can click on the trash bin icon to delete the move/assignment feedback with a line-through effect.

Add Students Name as Placeholder for Move Feedback The placeholder feature enables teachers to put real student names the feedback. Currently, we only support the placeholder for student name slot. Later the name will show up in the move feedback to give students more amiable experience. For example, the students might receive feedback from instructors like "Move up Jane!", "Keep Up John" in the student assessment page.

Select Type Teachers will be able to choose the four types of move feedback for each move including the correct response, the incorrect response, the neutral response and the close response in the radio button. For the assignment feedback, teachers will be able to choose among the proficient performance, the developing performance, and the basic performance. The default type for move feedback is neutral response and the default assignment feedback is developing performance.

History feedback with Emojis As sage is targeted for young students we added some emojis for feedback of different types and these emojis will then show up in the right panel of pop-up window consistent with feedback. Later these emojis will show up in the student assessment panel to add pleasure to game.

Default move/assignment feedback If instructors do not input any feedback at game level nor quest level, a set of default feedback is stored in the feedback service module under StudentAssessmentController.js.

5.1.2 Data Models

From sage project, we learned that each mission includes several quests and each quest comprises several games. However, two missions can have quests in common. The relation between mission and quest is many to many. The relation between the quest and the game is one-to-many. Based on this, we come up with assignment/game data model and quest/course data model with some modifications have been made to the previous data models. We store information related to a single game into the assignment data model and information related to a single quest to course data model. The quest ID will serve as a foreign key when linking games to quests. As can be seen from figure 6, the mission feedback could not be passed to quest level as multiple mission could have several quests in common. At the quest level, the quest will not be able to decide which mission feedback to be inherited from. The relation between quests and games will be one-to-many which means each quest could have multiple games but one game could only belong to just one quest. Then we come up with the following data model as is shown in figure 7, the foreign key would be quest ID. These two data model will help feedback data and CT Concepts data persist between the quest and game.

We store dummy data for move feedback in the moveFeedbackTemplate Model shown in figure 8. The templates will be wrapped in object with attributes type and comment string. It is noticeable that in the future the data model might be changed to add CT Concept attributes to relate feedback template to focuses.

5.1.3 Data Collection, Read, Update and Delete Pipeline

The data for feedback will be collected in input and stored in an array and add the type for each move/assignment feedback to become a JSON object. Through API, the feedback data is then sent to the course/assignment relations. For a read operation, each game/quest can read feedback on their own level. In addition to this, games will have access to retrieve the data from the quest level if it belongs to that quest. For update operation, teachers will be able to use the new data to update the old ones in the assignment relation. For the delete operation, teachers will first delete the data in the assignment relation and then use the remaining data as new data to update the old ones in the

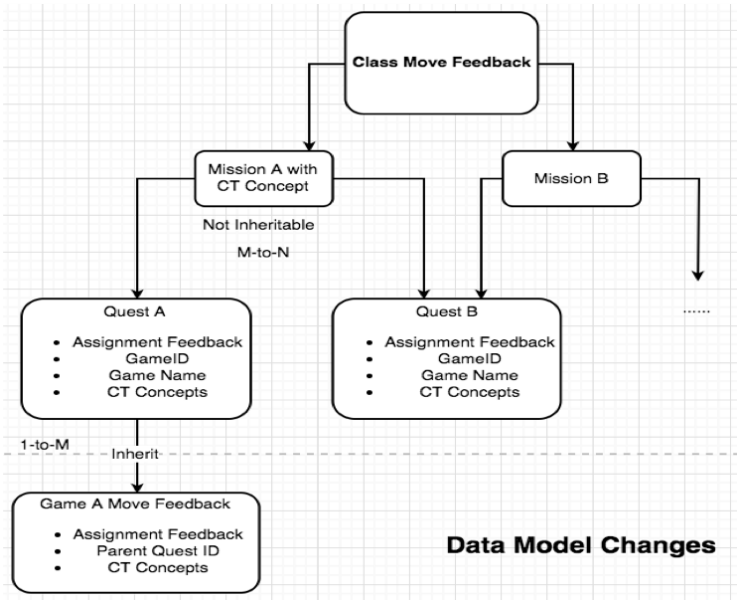

Figure 6: Data Model

relation. With successful deletion, the teachers will be able to see the updated remaining feedback in the right panel of the updating windows with a callback function.

5.1.4 Services

There are some services that we believe would be reusable for future routes.

Create a new assignment The parameter for this service would be all values containing the necessary information for creating a new assignment.

Remove an assignment from quest The parameters for this service would be course Id and assignment Id

Update course/quest general information This service will update the general information including course name, course description, focus areas. The parameters for this service are updating the values of attributes.

Update class level move feedback The parameters for this service are class Id and new moveFeedbacks values.

Remove an assignment from quest The parameters for this service are course Id and assignment Id.

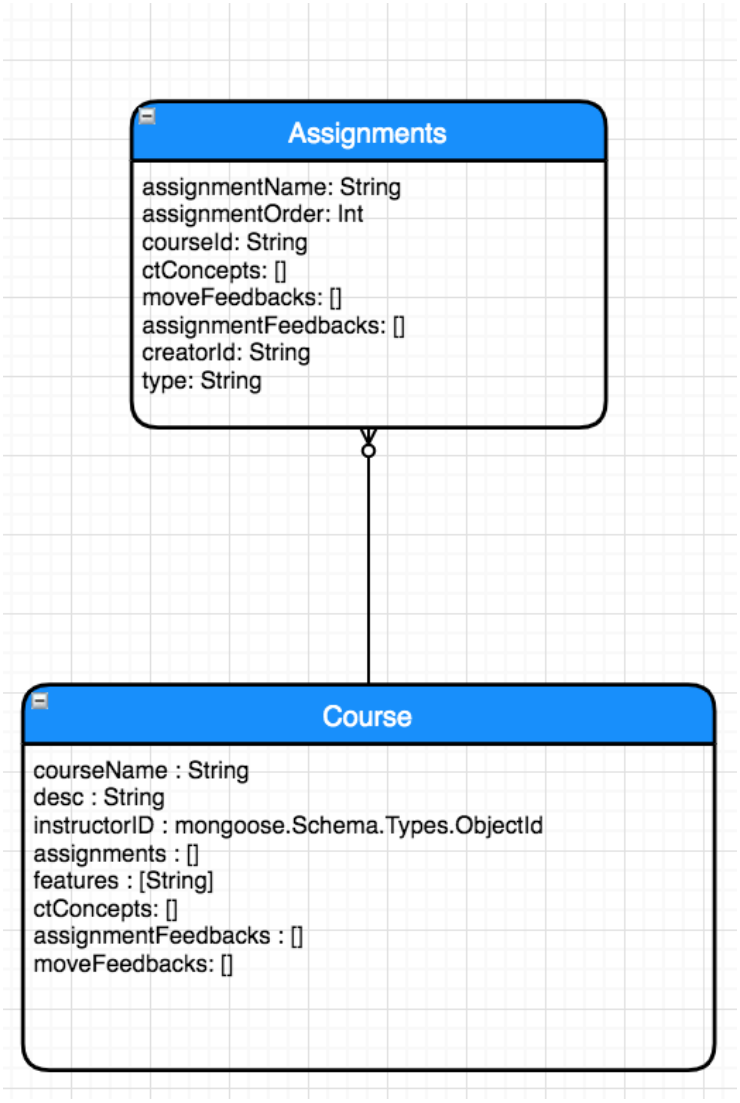

Figure 7: Relation between Assignment/Game and Course/Quest

5.1.5 User Interface

As is shown in figure 9, teachers can put move and assignment feedback and see this feedback shown under the input box after they confirm the feedback. They will be able to delete the feedback using the delete icon. They can send up to five comments of feedback to the database with the selected type. Then the feedback and corresponding type will be wrapped as JSON to store in the database.

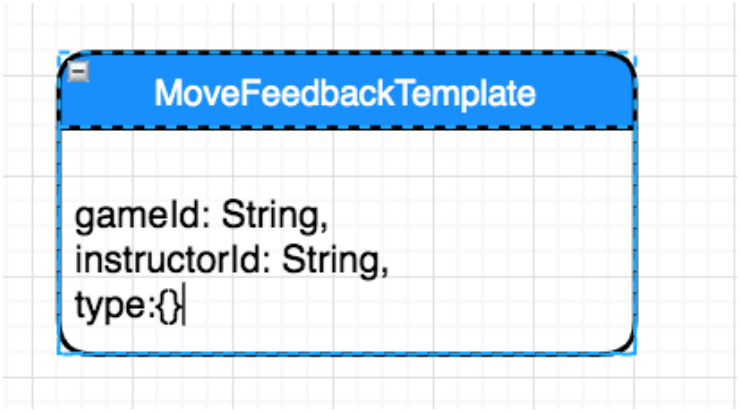

Figure 8: Relation of Feedback Template

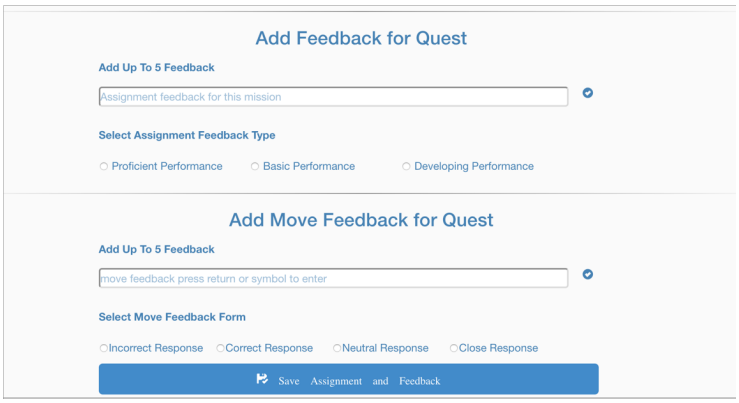

Figure 9: Move/Assignment Feedback Creation for Quest/Game

The updating assignment/move feedback will be operated in the pop-up window once teachers click the nav-bar on the top of the quest and game management page as is shown in figure 10.

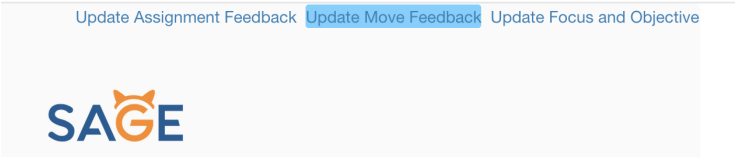

Figure 10: Click to Update Feedback and CT Concepts

As is shown in figure 11, in the quest level, teachers will be able to input quest level assignment feedback to update the quest level assignment feedback.

Using the left panel, teachers would be able to put more feedback comments to the database which will be reflected immediately in the right panel after they click the save button. The right panel of this pop-up window shows the current feedback added so far. This is the real-time reflection of the assignment feedback in the database. Once the teacher decides to delete assignment feedback at the quest level using the icon, the line-through effect indicates that the current feedback comments would be deleted. One click the trash icon, the current feedback comments together with its type would be deleted, the send an ajax request to the database to first delete the current feedback comment then update the feedback related to this request using the remaining feedback comment. The right panel will reflect the feedback change instantly.

In the game level, teachers will first see if the game level feedback is empty if it is then it will go one level up to get the assignment feedback from the quest level. Teachers will able to add, update and delete the assignment feedback at the game level and these data will be stored, deleted and updated in the database and these changes would be reflected immediately in the right panel.

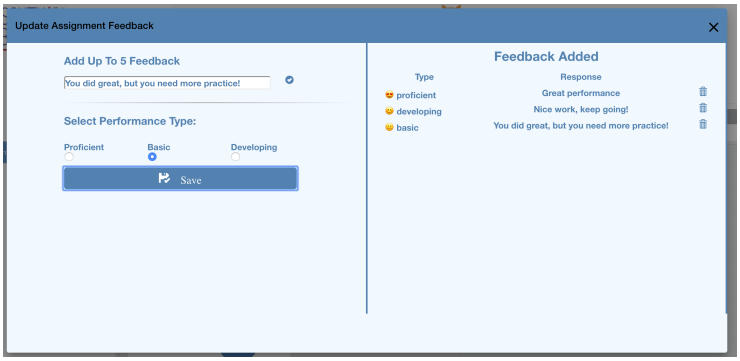

Figure 11: Update Assignment Feedback in Quest/Game Level on the Pop-up Window

As is shown in figure 12, for the move feedback at quest level, teachers would be able to input up to five feedback comments to an array and then wrapped those comments together with move feedback type to and store these objects to the database. Teachers can input student's name in the placeholder and this name will later be linked to the student in the student assessment page. Students will be able to see the move feedback together will their names on the game page. This feature enables a more amiable gaming experience as students can have more active interaction with teachers in the back of each game. In the game level, teachers will first see if there is any game level feedback. If not, they will be able to retrieve the quest level move feedback and inherit the move feedback from the quest. Teachers will be able to add and delete new move feedback by saving and deleting first followed by updating feedback using remaining data. Those feedback changes will be reflected instantly in the right panel.

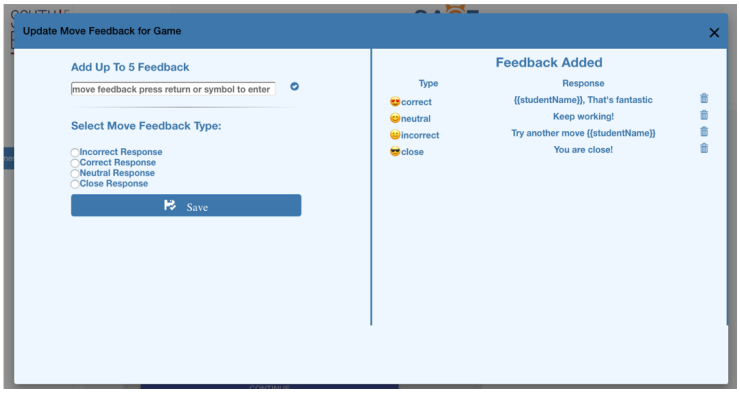

Figure 12: Update Move Feedback in Quest/Game Level on the Pop-up Window

To save teachers' effort further we offered them options to choose the feedback template stored in the database and this feature is not fully developed with CT Concepts but only dummy data so we disabled it from the development branch but routes and code are still in instructorGameController.js and scratchDesign.html. As is shown in figure 13 and figure 14, teachers can choose the template based on the type of feedback.

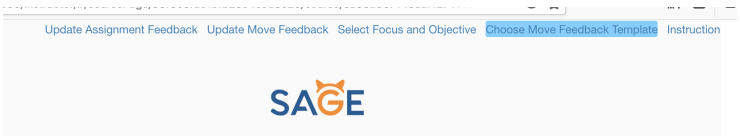

Figure 13: Choose the Existing Feedback Templates Nav Bar

5.1.6 Code Design

As the logic is similar when it comes to updating the move/assignment feedback for quest and games, also there are many places in views to update/create feedback, although the more common method here is to create a function as the controller for md-dialog in the parent controller file, we make a decision that we isolate the md-dialog controller function into one single file. InstructorUpdateMoveFeedbackController.js and InstructorUpdateAssignmentFeedbackController.js. By doing this we can significantly reduce the amount of duplicated code and make the code highly maintainable (now people only have to change one file instead of 4).

We then injected the controllers and views to the quest management page(instructor_questManagementContr for updating feedback at quest level, and instructorGamesController.js for game updating page.

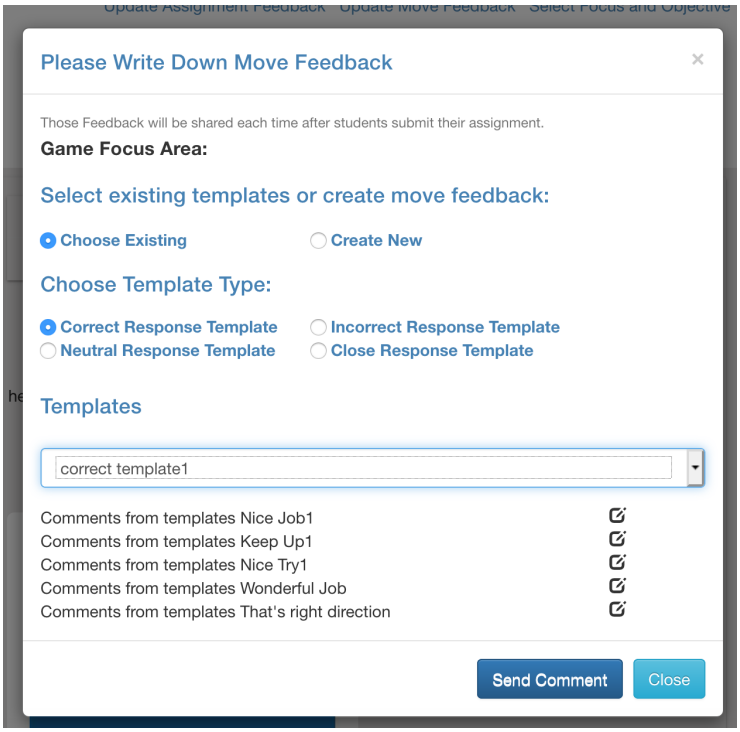

Figure 14: Choose the Existing Feedback Templates

5.2 Computation Thinking Concepts Labeling System

This system allows teachers to put CT Concepts at mission level, quest level, and game level.

5.2.1 Data Models

We store the CT Concepts in a JSON object with three levels and store the CT Concepts together with each game and quest.

5.2.2 Services

There are some services that we believe could be reused for future routes to operate on CT Concepts.

Get CT concepts from its parent quest level The parameters for this service would be game Id and course Id.

Update CT concepts for course The parameters for this service are course Id and new CT Concepts These services are reusable in many routes and could

be used for future work.

5.2.3 User Interface

In the mission level, the user will be able to select the focus of CT Concepts for each mission. They can reset and save the focus in a tree-like structure to the database as is shown in figure 15. This structure allows teachers to unfold each level and select and deselect the CT Concepts.

Figure 15: Feedback Creation for Mission

In the quest level, teachers will be able to create focus using a pop up window and store selected focus in the related quest as is shown in figure 16

As is shown in figure 17 at the game level, first we retrieved the quest level focus and show the focus for the quest with colorful tags: steel blue tags for primary level focus, light blue tags for second level focus and green tags for detailed level focus. Then teachers will be based on the quest level focus to create the game level focus. We believe that these tags and CT Concepts will be useful when classifying games by CT Concepts and offer student enhance training on certain topics and focus.

5.3 Student Avatar System

5.3.1 Feedback Presentation

Moved the student move feedback from sage-scratch into sage front-end. It is the goal of user story #266.

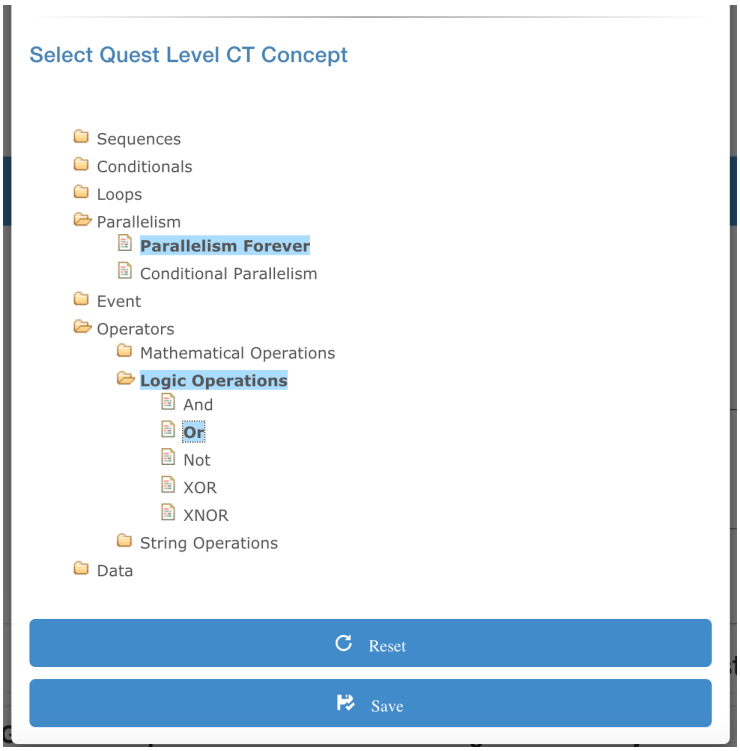

Figure 16: Create CT Concepts in Quest Level

5.3.2 Hint/Questions presentation

Just as we present feedback. We also make the hint and question presentation into the dialog box. So that student will not see a pop-up box anymore. User story #267 #268.

5.3.3 Time reminder

We include a small feature in the avatar system. The avatar now is able to remind how much time does the student left by sending a message to the dialog box.

5.3.4 History Statistics

During the semester, we also came up with an idea that can give student more information after they submit an assessment. We implement the routes and services so that the student will see their history performance. For example, a student can see how many time he has completed this assessment and how much score he/she got for previous sessions. This way, it may encourage students to do better next time. Apart from that, we can relate this with the CT concept

Figure 17: Update/Create CT Concepts in Game Level

that tells students how many points he/she gained under certain CT concepts. However, we did not make this feature show up in the development server as we think this function is not mature enough. We will discuss in detail what do we planned to do with this feature in the Future Work section of this report. The routes and service have been implemented and tested, so people after us

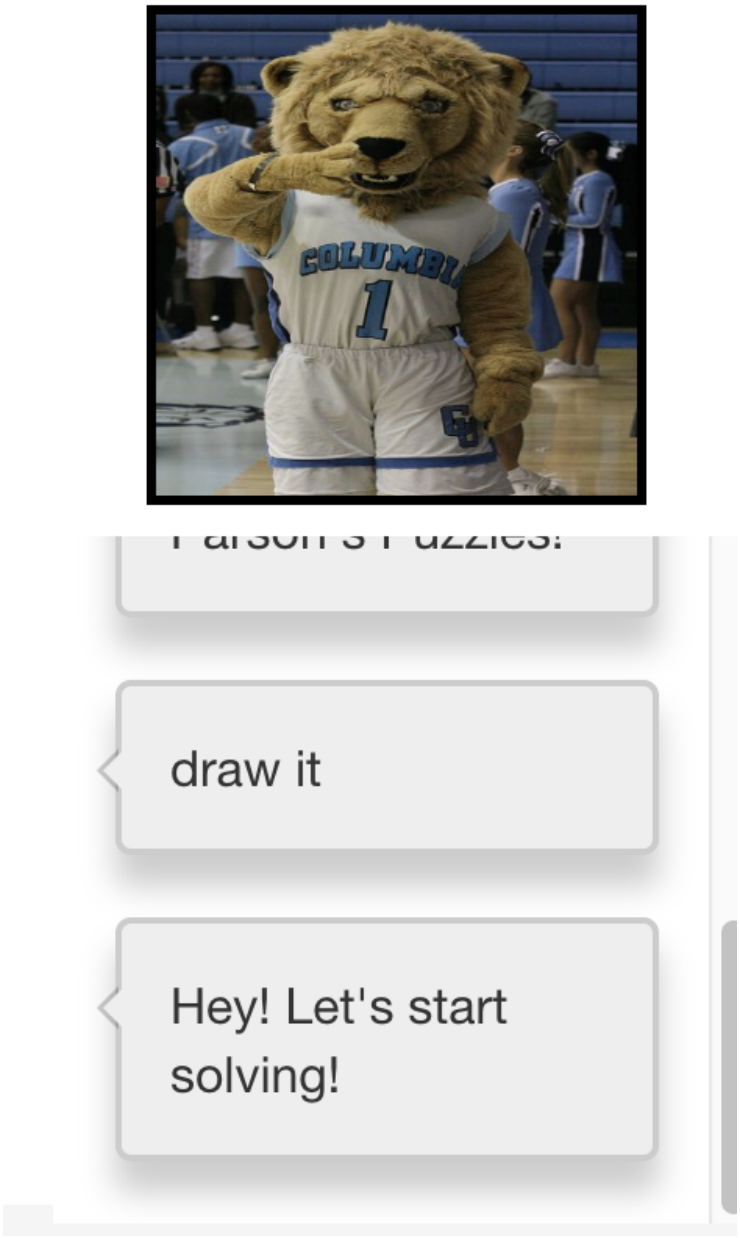

Figure 18: Feedback presentation

can pick it up really quick.

5.3.5 Feedback System

We build a highly maintainable feedback system for the sage-frontend to provide appropriate feedback for students. We structure the code in StudentAssement-Controller.js so that it is highly maintainable.

One of the main features of our feedback system is the hierarchical structure. This structure can give the instructor maximum ability to customize as well as the maximum efficiency to provide feedback for their students. On the student side, we are going to use this structure to provide proper feedback to students.

Since we made the assignment to be an isolated table and include a parent quest ID in each game's documents. We reduced many database lookups operations. During each game, the affinity space will first look up the move feedback and assignment feedback as the game level. If instructors have never input such feedback. Then the system to go up by one level and look up quest level feedback using the questID in the game database. The same logic used for getting assignment feedback. The process is demonstrated in the following figure.

Sage front-end will first preload all kinds of feedback (4 kinds of move feedback and 3 kinds of assignment feedback) into local storage upon student start an assessment. This is to reduce the latency of between student makes action and receive feedback. The feedback is then re-structured and stored as a dictionary in the front-end. Every time when students make an action or submit the assignment. Front-end will use a random selector to choose feedback from the local storage.

After select one feedback from the dictionary, the feedback service will further process it one more time before sent it to the screen. That is the placeholder feature. The placeholder feature is a feature that enables instructors could use some placeholder in their feedback. At the current stage, only one placeholder is supported, which is "student name". The placeholder will be replaced by real values before sent to the student. For example, as shown in the following figure 19 and figure 20, instructor input a string with a placeholder in it. When students receive this feedback, sage front-end will automatically detect such placeholder and replace it with student name. This way, we could let the student feel the conversation more inter-personally and interactive. In the future, we can add even more types of placeholders such as class name and so on.

After placeholder has been replaced with values, the feedback will finally push to the student side screen.

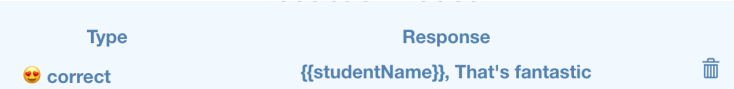

Figure 19: Instructor side

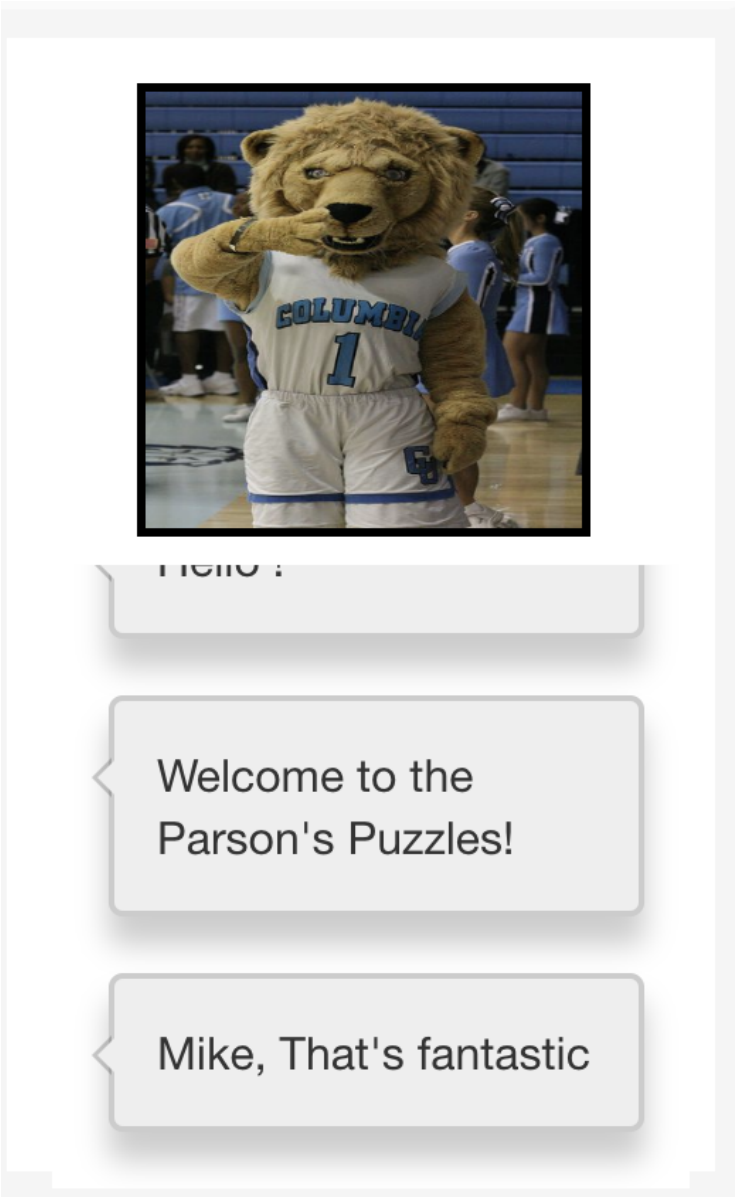

Figure 20: Student side

5.3.6 Code Structure

To make people who want to develop student side feedback service easier. We designed the code to make all feedback related services logic into one AngularJS service module. This module includes everything from retrieving from the

database, select placeholders, and final processing. The service module is then injected into the student assessment controller. This way, we could reduce the number of the logic happens with scope variables and give the code a better structure that is easily maintainable.

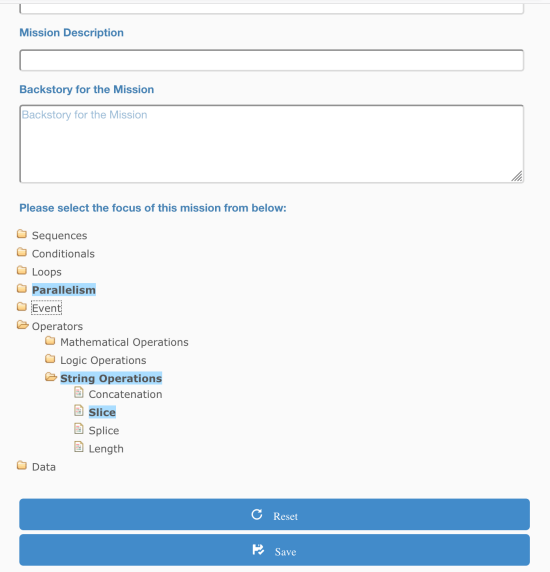

Figure 21: Instructor side

6 Bug Fixes

6.1 Learning path that student cannot receive feedback

When adding a new mission/learning path to a class, students cannot receive any updates. We fixed this bug.

6.2 Enrollment Issues

Instructor was not able to enroll a new student on its class management page. Meanwhile, even if student enrolled in to a class, student cannot see an updated list of enrolled class. We fixed this error.

6.3 Instructor class dialog box problem

The instructor class dialog did not always pops up and we fixed this problem.

6.4 Curricula focus styling and content

The focus in the mission page is not well styled. Before the page is look like figure 22

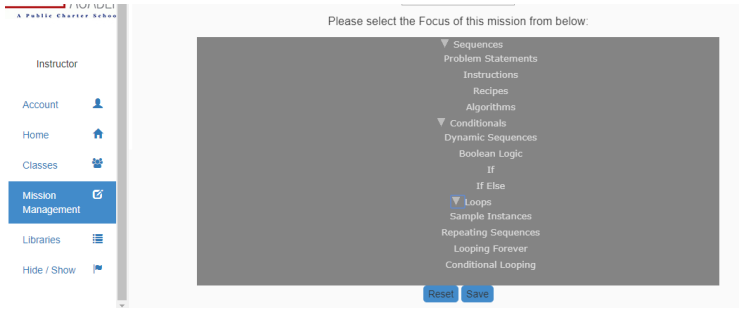

Figure 22: CT Concept page before in the Mission Level

6.5 Mission creation page styling and CT Concept Tree showing and hiding problem

The focus tree in the mission did not always show the CT Concepts and we fixed this bug.

6.6 Quest objective styling and hidden content

The quest objective page used to hide the content below, and the styling is really bad, we moved the three button to one level and add icon to the save button and we added the default route.

7 Future Work

7.1 Feedback Templates System

Although we have to build a system that uses a hierarchical structure that has significantly reduced the workload of the instructor when editing those feedback. We think we could take a step further and build a template system. Such template system should be persistence in the database. The instructor should be able to choose to use, edit or delete the template when composing the feedback for their students. Such a system should also provide a template based on the CT concepts. For example, we could potentially have one template that works well with games that focus on "Sequences". Instructors will then just need to confirm if they want the suggested template show up in this game/quest.

However, one should think about the relationship between instructor, game, and template. For example, should we allow instructors to edit and save a

new template? If so, should other instructor have access to such a template? If we allow templates to be shared with instructors, how should we define the boundary between the instructors? Should we change the data model so that each instructor is associated with a school? Apart from that, questions such as how should we design the feedback template to be related to the CT concepts is another big problem. Those are the question that needed to be considered before going into this direction

7.2 More CT Concepts

We extended the CT Concepts into three levels based on the papers mentioned in the related work. As there has not been an agreement on CT Concepts as researchers Mitchel Resnick and Karen Brennan mentioned in *New frameworks for studying and assessing the development of computational thinking*" Although computational thinking has received considerable attention over the past several years, there is little agreement on what a definition for computational thinking might encompass (Allan et al., 2010; Barr Stephenson,2011; National Academies of Science, 2010)"[6], there is still room for future students to dig more into CT Concepts in Computer Science field. Future students might design a questionnaire or reference more papers to extend CT Concept Scope.

7.3 CT Training System

As we have linked the CT Concept with each game, quest, and mission, we believe that CT Concepts could be used as criteria to classify games and a CT Training System could be designed to give students game modules that could be used to train specific CT Concepts. For example, some scratch games could be used to train the student's recursion, some games could be used to train iteration, and some games could be used to train their understanding of data types. Students will enhance their understanding of specific CT Concepts by doing extra exercise focused on specific CT Concepts. We believe such a system with games in modules with certain target and CT Concepts could help students achieve better learning result.

7.4 Student Submission Presentation

During this semester, we did not make too many changes on the pop-up models that show up when student submit their assessment. We only add the assignment feedback block, shown in 23. However, we think there is a lot of room for improvement in this section. For example, we could let the student know what are the CT concepts of this game, what concepts they have seen before but practiced again, what concepts are the first time that shows up, etc. To implement this, we would need to think carefully about how should we make students get enough information while not distracted too much. We should also think about how should we keep track of student progress in terms of CT concepts learning.

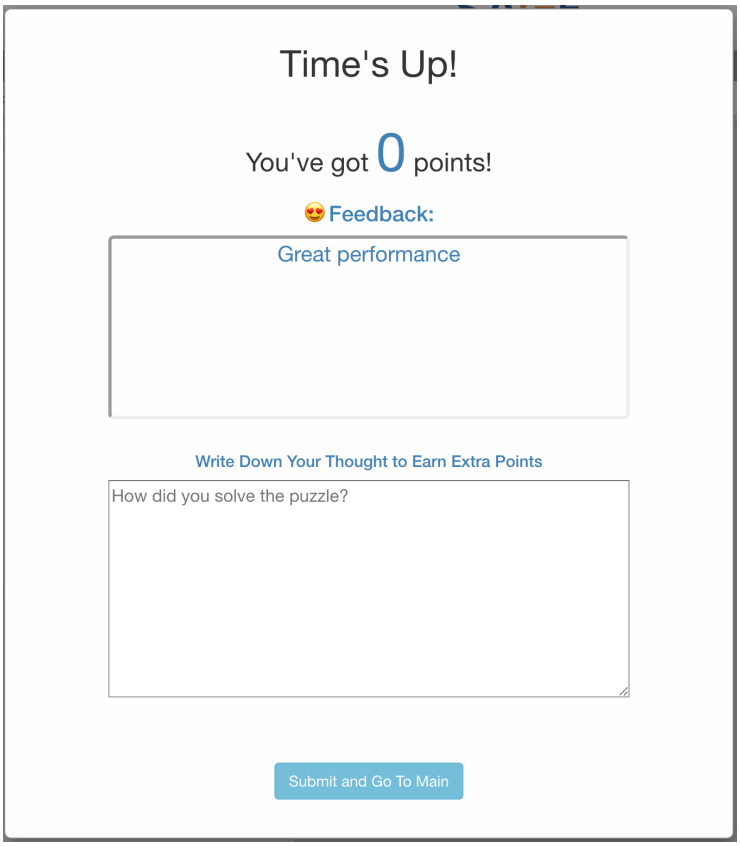

Figure 23: Current pop up model after submission

8 Conclusion

Our objective during this semester was to improve the Parson's Coaching system. We built a feedback system that not only highly customizable but also very efficient. The system also takes care of the different situations by dividing the feedback into two categories: assignment feedback and move feedback. Also, we extended the CT Concepts and link the CT Concept to game and quest level. We improved the user interface both in the student side and instructor side to offer students more guidance and bring pleasure at the same time. We structured our code base so that people after us could further enhance the features that we built very easily. Apart from that, we enhanced the system by fixing a number of bugs including user interface bugs, routes bugs, and service bugs. We also add a number of error handlers when making http calls and database calls.

References

[1] *About Scratch.* Scratch - Imagine, Program, Share. Web., 20 Dec. 2016.

[2] *CSTA K–12 Computer Science Standards.* The CSTA Standards Task Force, 2011.

[3] *CREATIVE COMPUTING: a design-based introduction to computational thinking Draft.* Friday, September 23, 2011.

[4] J. Bender. *Tooling Scratch: Designing a Collaborative Game-Based Learning System to Infuse Computational Thinking within Grade 6-8 Curricula.* Columbia University, New York, 2015.

[5] Michelle Chung Karen Brennan, Christan Balch. *CREATIVE COMPUTING.* Harvard Graduate School of Education.

[6] Mitchel Resnick Karen Brennan. *New frameworks for studying and assessing the development of computational thinking.* Paper presented at annual American Educational Research Association meeting, Vancouver, BC, Canada, 2012.