April 2015

ASSISTments Smart Reporting

Keaton Robert Smith
Worcester Polytechnic Institute

Yan Yan
Worcester Polytechnic Institute

Project Number: NTH - AAPZ

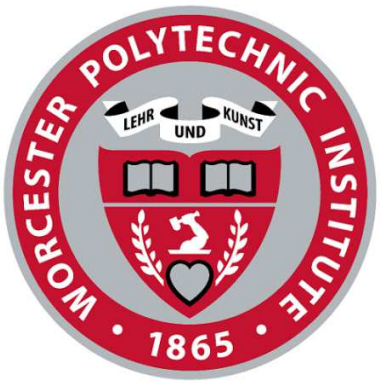
ASSISTments Smart Reporting

A Major Qualifying Project Report:

Submitted on April 30, 2015

To the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE



WPI

In partial fulfillment of the requirements for the Degree of Bachelor of Science

By:

_____

Keaton Smith

_____

Yan Yan

Advised by Professor Neil Heffernan

Abstract

This project extends the existing ASSISTments system for online learning to provide better feedback information to teachers using the service. Specifically we set out to design a new reporting mechanism so that teachers can see better information regarding a class's performance on a particular assignment. To achieve this goal, we implemented a new smart reporting scheme which allows for customization and better display of various types of assignments in ASSISTments.

Acknowledgements

We would like to thank Professor Neil Heffernan for his support and assistance throughout this project. We also would like to extend our gratitude to David Magid, Cristina Heffernan, Andrew Burnett, Chris Donnelly, Doug Selent, Yang Lu, Korinn Ostrow, and the rest of the ASSISTments team at WPI for their continual assistance and support throughout this project.

We also acknowledge funding for ASSISTments from the NSF (1316736, 1252297, 1109483, 1031398, 0742503, 1440753), the U.S. Department of Education GAANN (P200A120238), ONR's "STEM Grand Challenges," and IES (R305A120125, R305C100024).

Contents

Abstract 2
Acknowledgements 3
Contents 4
Table of Figures 5
1. Introduction 6
2. Background 7
2.1: The ASSISTments service 7
2.2: Current reporting system 8
2.3: Motivation for a new report 11
3. Methodology 15
3.1: Requirements 15
3.1.1: Report customization 15
3.1.2: Backwards compatibility 16
3.2: Development 17
3.2.1: Build environment 17
3.2.2: Integration with the existing ASSISTments system 18
3.2.3: Project design 18
4. Results 21
4.1: Smart reporting mechanism 21
4.2: Documentation 23
5. Future Work 25
5.1: Expansion to more report types 25
5.2: Enhanced report interface 25
5.3: Full system integration 26
5.4: Allowing users to tag reports 27
5.5: Implementation of other tagging 27
6. Conclusion 29

Table of Figures

Figure 1: A current teacher item report 9
Figure 2: A current teacher skillbuilder report 10
Figure 3: Section structure of a research problem set 12
Figure 4: Completion report for a research problem set 12
Figure 5: Item report for a research problem set 13
Figure 6: Web GUI to specify sections to report on 22
Figure 7: Web output of a smart report 22
Figure 8: A current report in ASSISTments 25

1. Introduction

The field of education is increasingly moving toward online platforms. With the development of various technologies to enable learning in a web atmosphere, teachers and students can take advantage of data, services, and educational platforms across the internet. Such online tools require continual development and improvement to achieve the goal of bettering the educational process for everyone involved. This means analyzing user data, determining the needs of students and teachers, and utilizing the full capabilities of computer systems to provide the best possible feedback during the learning process.

One such service is ASSISTments, an online tutoring tool which has been in development at the Worcester Polytechnic Institute since 2003. ASSISTments aims to provide useful information to not only teachers and students but also parents, school administrators, and educational researchers to facilitate better learning in and out of the classroom. This means that the developers of ASSISTments are largely concerned with logically and simply presenting the data at their disposal to anyone involved in the system.

One such mechanism is a report for teachers which displays information on a class's performance on a particular assignment. The improvement of these reports is the focus of this MQP so teachers can have a better understanding on how their classes learn best. We explore this relationship by analyzing the current reporting mechanism for teachers and outlining requirements for an improved reporting scheme. Next we detail our implementation of this scheme and analyze it based on the original requirements and integration into ASSISTments. Finally we discuss plans for future upgrades and extensions on our system.

2. Background

This section of our paper details the background knowledge we needed to know before beginning our project and serves to set up the rest of this paper. We begin with an overview of the ASSISTments service and its philosophies so we can keep in mind the overall goals of the system throughout. We next detail the current structure of reports for teachers, and finally move on to the motivation for an overhaul of this system, which becomes the crux of our project.

2.1: The ASSISTments service

ASSISTments is a service being developed at Worcester Polytechnic Institute designed to improve the educational process by discovering how students learn best. It is primarily an online tool used by teachers to distribute content for their students, especially homework, quizzes, and tests. ASSISTments is based on the principle of providing feedback to students as they work, as studies have shown that this makes the learning process more effective. Such feedback in ASSISTments can take the form of educational videos, written hints on a problem, or reports on assignments as a whole.

As such, within ASSISTments there is the capability for teachers and educational researchers to develop content for students to use. This content can then be shared across the entire ASSISTments user base, and other teachers and researchers on the site can assign it to their students or modify it to suit their needs. From a teacher's perspective, this makes it easy to ensure students are learning according to the appropriate curriculum while also providing methods to avoid reinventing the wheel when developing assignments for students. From the perspective of the educational researcher, the large user base and

capability for content sharing within ASSISTments delivers a useful platform for conducting studies regarding how students learn best.

In regard to the content which can be distributed using ASSISTments, there are multiple different types of problem sets which can be built by teachers and researchers. These sets begin with simpler styles, which take a group of problems and present them to students in either a set or randomized order; these problem sets are known as linear sets or random sets, respectively. Skillbuilder problem sets are an extension of these two types which require a student to get three correct answers in a row to mark the set as completed.

There is also support for problem sets which randomly assign each student to do one of several problems or sections (called choose conditions), or sets which deterministically assign students to specific content based on their answers to an initial problem or set of problems (called an if-then-else conditional). These two types of problem sets might be used by researchers who want to determine which of several techniques best facilitates student learning.

2.2: Current reporting system

Because of all the flexible content in ASSISTments, and keeping in mind that the system is largely data-driven in its philosophy, there is a necessity for ways to report that data to everyone involved in the process—from students to researchers to parents to teachers. Throughout this project we focus primarily on the reports given to teachers regarding class-wide assignments.

Currently the teacher reports include two primary types of reports, each used for different types of problem sets. The first report we will discuss is known as the item report,

which provides a problem-by-problem look at the performance of students in the class on a particular assignment. See figure 1 below for a screenshot of an item report of a standard assignment.

Student/Problem [Anonymize]	Average Data driven	788357 Data driven	362730 Data driven	362729 Data driven	362733 Data driven	362738 Data driven	362725 Data driven	362726 Data driven	362727 Data driven	Total Hints	Time Spent
Problem Average Graph	38%	100%	100%	0%		100%	100%	100%			
Common Wrong Answers											
Correct Answer(s)		*hi*	*499*	*782*	*957*	*272*	*237*	*897*	*864*		
Smith, Joe	0%									0	N/A
Smith, John	13%	✓ hi	◐ 166.							0	00:03:03
Smith, Keaton R	63%	✓ hi	✓ 499	✗ 23		✓ 272	✓ 237	✓ 897		0	00:03:17

Figure 1: A current teacher item report

Each cell in the item report represents how a particular student did on a particular problem. It contains information including the answer the student gave and a symbol indicating correctness and completeness of the problem. Each column represents a single problem in the problem set as a whole, while each row represents a student. There are also cells which give the teacher information about class-wide data on each problem, including percent correct, possible correct answers, and potentially some common wrong answers given by students. The item report is currently typically used in problem sets which have a linear or random structure, since it is guaranteed that every student will see every problem and so the teacher will be able to easily make sense of the report.

The other main type of report currently available in ASSISTments is the completion report, which in general provides information to teachers on an assignment level, as opposed to reporting on individual problems. There are several variations on the completion report, each providing different statistics to the teacher. As an example, consider one such variation, the skillbuilder report, in figure 2.

Student [Unanonymize]	Status [Show dates]	Days working on assignment	Total time working on assignment	Problems seen
XXXXX	✓	1	00:41:19	4
XXXXX	✓	2	2 days 08:56:51	7
XXXXX	◐	1	00:12:15	2
XXXXX	✓	1	02:19:41	4
XXXXX	✓	1	01:12:03	8
XXXXX	✓	2	13:21:19	13
XXXXX	✓	1	00:49:12	5
XXXXX	✓	1	00:07:32	3
XXXXX	○			
XXXXX	✓	1	01:13:07	6
XXXXX	✓	1	00:32:27	4

Figure 2: A current teacher skillbuilder report

Again, in a completion report, each cell represents information on a particular student's performance. However, in this case that performance is measured with certain statistics for the entire problem set, each of which is assigned to a column in the report. Such statistics vary based on which style of completion report is being used, but in general can include completeness of the problem set, average correctness across all the problems, time spent on the assignment, and number of problems encountered by the student. Contrary to the item report, the completion report is typically employed by the system when reporting on assignments in which it is not guaranteed that every student will have encountered every problem, and so would not make sense to report on every problem from the teacher's perspective. Examples on such problem sets include research studies performed with if-then-else or choose conditions, or skillbuilder sets, which contain dozens of problems but typically only show a fraction of those to each student before completion is awarded.

The current reporting structure in ASSISTments makes use of the ASSISTments web service to build and render assignment reports for teachers. This process is done client-side whenever a teacher asks to see a report for an assignment. Upon accessing the

ASSISTments database to retrieve relevant information, the web service uses Ruby on Rails to consume the data and generate an appropriate HTML webpage to display the data in the form seen in figures 1 and 2. It is worth noting that the ASSISTments web service is a separate body of code than the ASSISTments API, a library of classes which makes up the server-side architecture of ASSISTments and are accessible over HTTP.

2.3: Motivation for a new report

However, the current system has some shortcomings, especially in dealing with research problem sets which incorporate if-then-else and choose conditions. As an example, take the research problem set displayed below in figure 3. This is a randomized control trial which aims to determine the effectiveness of video hints versus text hints in facilitating student learning, and as such has multiple branches of problems—some with video and some with text. In this case, currently the current system defaults to a completion report (see figure 4) which gives basic information about the problem set. But when compared to the complex structure of the research problem set these reports result in the loss of a great deal of information which might be useful to the teacher.

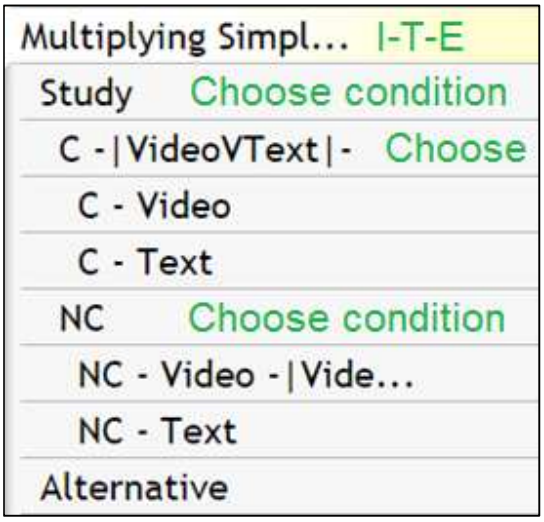


Figure 3: Section structure of a research problem set

Student [Unanonymize]	PSATZEJ ---- 70 - Multiplying Simple Fractions 5.NF.B.4a EX
Average	79%
Complete	22%
XXXXXXXX	○
XXXXXXXX	○
XXXXXXXX	100%
XXXXXXXX	○
XXXXXXXX	75%

Figure 4: Completion report for a research problem set

There is also the capability in the current system for the teacher to gain more information than is displayed in figure 4. However, this information comes in the form of an item report which provides data for all students on all problems, even those which a given student would never see due to their placement within a choose condition. This results in information overload on the teacher, as such reports are confusing to parse and filled with unnecessary cells of data. To give an idea of the extent of this issue, an item report of the video versus text problem set is displayed below.

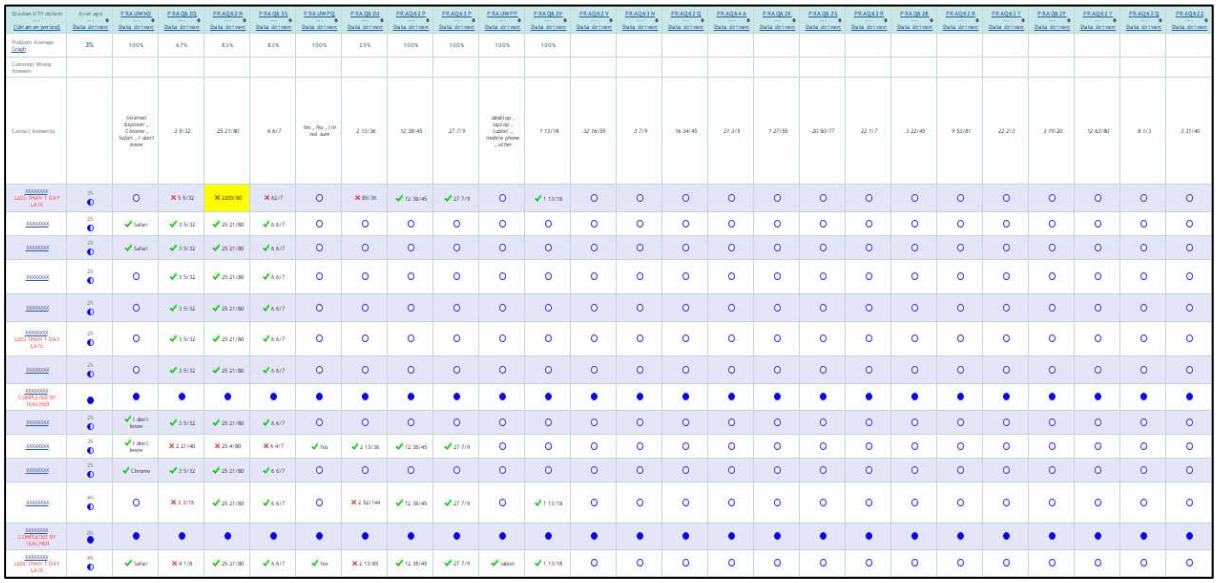


Figure 5: Item report for a research problem set

Clearly there is a need for a reporting mechanism which can capture some of the complex structure of research problem sets with multiple choose and if-then-else conditions. This report also needs to be understandable for the teacher, not rive with unnecessary data if students weren't presented with some of the same problems. These are the issues we tackle in this paper with our smart reporting mechanism, which aims to replace the current teacher reporting system within ASSISTments.

A secondary motivation for a new reporting system has come about due to recent expansions of the ASSISTments user base. These expansions come in the form of students and teachers looking to use tools provided by ASSISTments through a new protocol called ASSISTments Direct. This protocol enables students to complete assignments via links distributed by teachers without needing an account or logging in to the ASSISTments web platform.

To better service users of ASSISTments Direct, it is desirable to have a new reporting scheme which is modular and separate from the web service itself. Such a

separation would make it easier in concept for report information to be distributed to involved parties. As we will discuss later, our smart reporting mechanism is designed with this philosophy in mind.

3. Methodology

In this section, the methodology of creating smart reports will be discussed. Firstly we state the requirements which smart reports should fulfil, and then describe the actual development of smart reports. There we mention the setup and the build environment we used as well as the integration of the smart reporting with the existing ASSISTments system. Finally, we end with an overview of all the classes, data structures, functions, and algorithms we built to create the smart reports.

3.1: Requirements

In developing the smart reporting system, we were in close collaboration with the rest of the ASSISTments development team to ensure we were on the right track. Throughout this communication we came up with two main requirements for our system, given its goal to handle all types of problem sets. We wanted to include the ability for creators of assignments to customize their reports along with backwards compatibility with assignments created before the idea for our system. These are detailed below.

3.1.1: Report customization

As described above in section 2.3, the major motivation behind smart reporting is to be able to handle many different styles and structures of assignments in ASSISTments. As an initial solution to this problem, we looked at implementing a set of rules which would be able to ascertain the appropriate way to report a particular section of a problem set based on if every child doing the assignment would encounter that section at some point. However, after careful consideration we decided this solution had the potential to be

difficult to understand from a development perspective and so had a lot of possibilities for errors in reporting different assignments, especially with some of the more complex research problem sets. Therefore, as an alternative solution we decided to enable the builders of assignments and problem sets to customize how they were reported.

This approach left a lot potential room for errors in our code, as we weren't attempting to create some rules which would be robust enough to handle every use case of the system. It also created a lot of room for better information to appear in the reports for the teacher, as the builder of the problem set—the one who knew the structure the best—would get to intelligently decide what made the most sense to report on. For example, maybe some sections in the set were simply ascertaining basic information, such as whether or not the students could view videos. In this system, the builder of the problem set would know that and be able to leave that section out of the final report. Thus ultimately we decided that being able to pick and choose the style and information included in the report was more important than a set of rules which tried to predict the correct output.

3.1.2: Backwards compatibility

We were aware, though, that we could not solely rely on the builders of problem sets to tell us exactly how assignments should be reported. If nothing else, we needed to maintain the ability to report on any problem set in ASSISTments before our reporting scheme was conceived. With that in mind, we decided also to develop a limited set of rules which could robustly handle any problem set but might not provide all the information that a problem set builder would desire. After discussion with the rest of the ASSISTments team,

we arrived at this solution due to the fact that it would provide more information than the reporting scheme, and that the importance of displaying old assignments would become less and less important as time went on. We detail this default report structure more in section 3.2.

3.2: Development

In this section we detail our development process in building a smart reporting system. First we go over the environment we used to implement and test our code, and how that code needed to be integrated into the ASSISTments service. Finally we give an overview of all the code we wrote to build our smart reports and explain in general the purpose of all our functions and classes.

3.2.1: Build environment

The bulk of our development was done within the ASSISTments API, which is written solely in Java. To write this code we used the Eclipse IDE, configured to include copies of all the API code written up to the start of our project. We also installed a Tomcat server within Eclipse, which enabled us to test the portions of our code which required access to the ASSISTments database tables.

Furthermore, to simulate the HTTP request which would be used by a client to access the ASSISTments API, we used the Postman extension in Google Chrome. Postman is a tool which allows developers to send customized HTTP requests and view their responses for testing. We also used Eclipse to develop a sample client webpage to display our reports, written in JSP.

3.2.2: Integration with the existing ASSISTments system

An obvious necessity of our system was ease of integration with the rest of the code for ASSISTments. To facilitate this, we designed our code to be similar in structure to other files within the ASSISTments API. We followed existing protocols when adding capabilities to our reporting mechanism with respect to database accesses and responses to HTTP request to ensure there would be as seamless an integration as possible. Furthermore, throughout our development process and our additions to the existing ASSISTments codebase, we ran tests to ensure that nothing we were doing was interfering with the rest of the system.

Full integration with the ASSISTments web service is discussed in our future work later in this paper. For purposes of this project, in addition to developing within the API, we added a webpage to the ASSISTments Teacher Tracker (a separate service which provides information on users of the system) to display our reports. This we developed in much the same way as our API code, with an eye toward ensuring that our changes were both in the spirit of the system's architecture and not interfering with the functionality of the existing code.

3.2.3: Project design

As Java is an object-oriented language, the smart reports we generated are essentially Java objects, and the data they contain are also in the form of objects. In this section we talk generally about the process of utilizing these classes to create a smart

report. For a complete and detailed look into all the classes and files which are part of this project, see appendix A.

The process of creating a smart report begins when the ASSISTments API receives an HTTP request from a client who wants a report on a particular assignment. This request should be a POST request, which provides a reference to an assignment as well as an optional JSON string representing the desired reporting structure, known as a tagging object.

From this point, we use various classes to access the ASSISTments database to get necessary information, including the students who were given the specified assignment, the problem logs representing their answers to each question in the assignment, and the structure of the sections within the assignment. Using this information, we create a tree of sections and problems within the assignment. The leaf nodes of this tree represent individual problems, and it is at these leaf nodes that we store the information on student progress on the assignment.

Once this tree is created, we can then traverse it from the root section to build the smart report itself. It is at this point that we consider the optional tagging object; if one is provided, we form reports on different sections in the tree according to that object. Otherwise, by default we create an item report for each section of linear or random problems, and completion reports on choose conditions, if-then-else conditions, and sections of skillbuilders (as described in section 2.1).

After the tree is traversed, all the necessary reports for the sections within the assignment have been generated and added to a smart report which represents the report for the assignment as a whole. At this point, that smart report is converted into the JSON

format so it can be sent as a response to the original HTTP POST request. This JSON can then be used by the requesting system to display the information contained within the report.

4. Results

In the results section, the detailed architecture of the smart reporting mechanism is described, and the usage of smart reports is discussed. The documentation of the smart report system API is in section 4.2 for future developers to further extend the system.

4.1: Smart reporting mechanism

At the end of our project, we had built a smart reporting mechanism which met many of the goals we outlined at the beginning of our project. We built a system which can generate intelligible information about many complex structures of problem sets in ASSISTments and do so in a way that it does not include unnecessary amounts of detail. Our system has the capability for customization in reporting style through inclusion of a tagging object, in which different reports encompassing different sections of a problem set can be described. Our system's design is also created with global integration in mind in how we create and send data over HTTP, and thus could be easily integrated into new and future services like ASSISTments direct.

Furthermore, for users to better understand the usage of our smart reports, we also created a simple web user interface built on top of the current TeacherTracker project to display how input and output of the smart reports looks. This also allows the ASSISTments team to experiment with our smart reporting system and discover ways to enhance it prior to its full deployment onto the system. The following figures show the screenshots of this webpage.

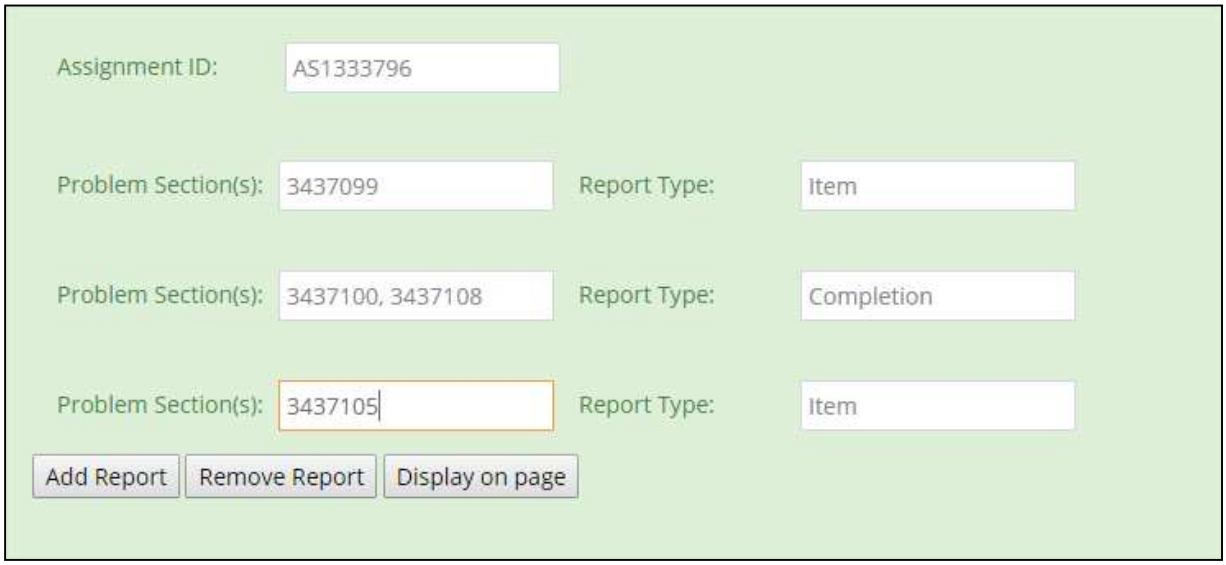


Figure 6: Web GUI to specify sections to report on

User ID	754934	Completion Status	Percentage Correct	Number of problems seen	Number of problems done correctly	Time taken
282463	Wpi (correct)	COMPLETE	100%	3	3	0:00:59
282476	WPI (correct)	COMPLETE	75%	4	3	0:02:08
282464	NOT STARTED	COMPLETE	33%	3	1	0:04:01
282470	Wpi (correct)	COMPLETE	100%	3	3	0:00:27
282469	novideo (incorrect)	STARTED	100%	3	3	0:02:39
282467	Wpi (correct)	COMPLETE	75%	4	3	9:27:33
282465	WPI (correct)	COMPLETE	100%	3	3	0:08:11
282473	WPI (correct)	COMPLETE	100%	4	4	0:03:17
282471	WPI (correct)	COMPLETE	100%	4	4	0:07:21
282462	Wpi (correct)	COMPLETE	60%	5	3	0:04:59
282468	WPI (correct)	COMPLETE	44%	9	4	0:08:33
282472	WPI (correct)	COMPLETE	100%	4	4	0:08:05
282466	Wpi (correct)	COMPLETE	100%	3	3	0:04:31
281092	NOT STARTED	NOT STARTED	0%	0	0	0
281091	NOT STARTED	NOT STARTED	0%	0	0	0
281089	NOT STARTED	NOT STARTED	0%	0	0	0
281097	NOT STARTED	NOT STARTED	0%	0	0	0
281096	NOT STARTED	NOT STARTED	0%	0	0	0
281467	NOT STARTED	NOT STARTED	0%	0	0	0
281645	NOT STARTED	NOT STARTED	0%	0	0	0
281245	NOT STARTED	NOT STARTED	0%	0	0	0
281225	NOT STARTED	NOT STARTED	0%	0	0	0
281471	NOT STARTED	NOT STARTED	0%	0	0	0
281468	NOT STARTED	NOT STARTED	0%	0	0	0

Figure 7: Web output of a smart report

In evaluating these results, we used two main metrics. First was our communication with the rest of the ASSISTments team, particularly Cristina Heffernan and Andrew Burnett, regarding the criteria they determined would be best from a teacher's perspective in the smart reports. In that respect, we were able to create a system which met many of those criteria in simplifying the data displayed to teachers as well as providing intelligible data for all types of problem sets. Due to time constraints, we did not meet some of the other eventual goals of the system, including full ASSISTments integration, persistent storage of each assignment's reporting scheme, and addition of more information and report types into our system. These are detailed below in our future work section.

The second evaluation method we used was in the modularity of the system's structure. For this we communicated largely with the chief software engineer on ASSISTments, David Magid, to ensure our code was developed in a way that supported future development and integration. By this standard our project performed well, as demonstrated both by our ability to separate the reporting structure from the web interface and the capability of our reporting mechanism to be used in service like ASSISTments Direct.

4.2: Documentation

To ensure future developers and users of ASSISTments are able to work well with our code, we both augmented our code with standard Javadoc-style comments and created or added to three external documents with relevant information about our system. The first of these is a document describing attributes of our tagging object so future developers can form similar objects to interface with our code. We also created a writeup describing

how to test our code using the user interface we developed. Finally, we added to the existing ASSISTments API specification to record the appropriate HTTP requests to send to our code when requesting a smart report.

5. Future Work

In this section, the potential improvements of different aspects of the smart reporting mechanism will be described, and directions of future projects that focus on the extension of our current system will be discussed.

5.1: Expansion to more report types

Currently the smart report can generate reports of three types, namely the item report, the completion report and the skillbuilder report. More report types such as the opportunity report and proficiency report exist in the ASSISTments system, which would be desirable to include in the smart reporting system as well. As the aim for designing the smart reporting mechanism is to provide flexibility in generating reports, and given the fact that our code is extensible in handling new report types, it is possible for future developers to extend our project to provide support for currently missing and newly introduced report types at the assignment level.

5.2: Enhanced report interface

All the reports displayed in the ASSISTments website share a common style, and one example is shown in the figure below:

Skill	Skill Grade	Sample Problem	Percent Correct	Ratio Correct	Students who need to remediate
[+] Division of Positive Decimals	6	PRAJJSH	0%	0/1	1
[+] Operations with scientific notation	8	PRADKB3	50%	1/2	1
Compare positive decimals	5	PRAJYYK	100%	1/1	0
Recognize statistical questions	6	PRANKP4	100%	1/1	0
[+] Center, spread and shape of data	6	PRANUNA	0%	0/1	1
Properties and Classification Quadrilaterals	5	PRAE4A4	100%	1/1	0
Combining Like Terms	7	PRAH3S4	100%	1/1	0
[+] Scale drawings	7	PRABVUZ	0%	0/1	1
Divide multi-digit numbers	6	PRAFDY3	100%	1/1	0

Figure 8: A current report in ASSISTments

As smart reports do not have the default style sheet used by the reports in the ASSISTments website, the style of the smart reports does not conform to the rest of ASSISTments system. Thus, one potential improvement of the smart reports could be to unify the reporting style to fit the current ASSISTments environment.

Furthermore, as previously described, we do provide various data regarding different aspects of students' performance and the problems themselves in the JSON string sent out as the output of our system. However, in our current web user interface, we do not display them to the user. One future project could focus on the better utilization of the underlying information provided by the smart reports to make the reports more informative and useful to teachers.

5.3: Full system integration

Currently our smart reporting system only serves as an external service to the ASSISTments website, such that it could only be accessed as an expanded feature external to assistments.org. As such, it is not available to users who simply log into the ASSISTments website and look for it in the drop-down menu showing all kinds of reports. As the smart reports is designed to provide help to the teachers in better understanding the performance of the students, it could be a future project to integrate our smart report system into the ASSISTments website and make it the default way to generate reports on the teacher side.

5.4: Allowing users to tag reports

Another future direction could be to store and reuse tags that are created as the input for generating smart reports. As in the current smart reporting system, the tagging of the problem sections is limited to a single use per reporting, and the user has to provide the same tagging mechanism each time if the smart report of certain format is needed for multiple times. Storing tagging information in the database could resolve this problem, and this could be done by adding tagging feature in the builder of a problem set, so that the users could create and edit tags when they build or modify problem sets and assignments. By doing so, when a smart report is requested, the relevant tagging information could be retrieved from the database to facilitate users in creating association between problem sections and report types.

5.5: Implementation of other tagging

As previously discussed, through tagging, problem sections coming from different origins can be aggregated together to better classify the sections based on different user-defined criteria. It is also possible, as a future project, to redesign the tagging system within ASSISTments on the basis of problem sections, so that the tags themselves could serve as an informative way of representing the nature of certain problem sections, and also could exist independently to the reports. In this manner, when the reporting of an assignment is to be done, tags related to the problem sections within such assignment could be retrieved to help teachers with better examining the report. However, the way of reporting would not necessarily depend on the tags themselves, as the teacher could choose the sections to report independently of how the tags are constructed. By doing so,

the tags could be used to assist teachers in associating certain problem sections with similar ones, but teachers could still have the freedom of customizing report with any combination in the problem section level if they wish.

6. Conclusion

The smart reporting mechanism as a concept and in our implementation has the possibility to provide a great deal of value to the ASSISTments platform. With the added information that our infrastructure can provide, teachers should be able to view statistics on student assignments in a more effective manner. Furthermore, the potential for customization of reporting structures in our system give more control to researchers and builders of problem sets, which in turn only makes the information feedback that much more valuable to teachers. Furthermore, our project is designed in a way which supports future expansion and modularity in the system, which only serves to heighten its potential for improvement.

We as students have also learned a great deal throughout the development of this project. Through it, we have gained real-world software engineering experience in development of a feature addition to a marketed system and so learned how to program to fit specifications and requirements. The knowledge we have gained in working with such a large and long-standing system as ASSISTments is also particularly valuable, as programming in such large environments is the core of software engineering. And in through all this, we believe that we have laid the foundation for a potentially valuable addition to the ASSISTments service.

Appendix A: Files added and modified

New Data Structures

SmartReportObject

Path: org.assistments.domain.SmartReportObject

SmartReportObject is the class that contains all needed information for the smart report, and is converted into JSON format before being sent as the response to the HTTP request. It contains a list of AbstractBasicReport as reports to display and functions to add an AbstractBasicReport to the list.

AbstractBasicReport

Path: org.assistments.domain.AbstractBasicReport

AbstractBasicReport is an abstract class that should be extended by any actual report type object, and it contains variables shared by all report types as well as the functions that should be implemented differently based on the type of report. Specifically, it contains a list of user IDs, a list of Section objects, and a string indicating the name of the report type. The list of user IDs and the list of sections are needed in the constructor of the AbstractBasicReport. It contains an abstract function named generateReport() to run the algorithm to create the report from all available data, and another abstract function named getReport() that returns a list of ReportObjectColumn that are useful in displaying the report.

Section

Path: org.assistments.domain.Section

The Section object represents a section of a problem set and a tree relationship between its sections. One Section object corresponds to one row in the section table of the ASSISTments database. Each Section object contains an ID, a type, a field to store the information regarding parent sections, a list of sections that represent all the children of this section in the tree, and a list of problem logs that are associated with this section. With this construction, it is easy for the developer to run an algorithm to process problem logs at the section level.

For creating the Section object, the id number of the section, the parent section and the type should be specified in the constructor; if the section has no parent, the parent parameter can be set to null. In terms of the functions, the Section class contains all getters and setters of all fields it has, along with a function named findByID(long id) to find a particular section by its ID number in this section tree, which is done by a breadth-first search and returns the found Section or the null value. It also overrides equals() and hashCode() functions to allow comparison between different Section objects.

Answer

Path: org.assistments.domain.Answer

The Answer class is also a wrapper to store information on a particular problem. It contains the correct answer or multiple correct answers of a given problem by having a problem ID field and a list of strings representing all correct answers of that problem. In order to create the Answer object, the problem ID is needed in the constructor. There are

getters and setters for both problem ID and the correct answer list to allow the user to modify the object afterwards.

TaggingObject

Path: org.assistments.domain.TaggingObject

The TaggingObject class contains information related to the choice of tagging problem sections by the user for report customization. This information is provided in the form of a JSON string when a report is requested over HTTP. This is then converted into the real TaggingObject object to allow further processing.

The TaggingObject class only has one field, a list of ReportTypeRow objects, which will be discussed later. For creating a TaggingObject, the list of ReportTypeRow could be passed in as the parameter to the constructor, and the constructor could also take no argument; in the latter case, the list will be initialized to contain no objects. Functions are provided to view and manipulate these rows within the object

ReportTypeRow

Path: org.assistments.domain.ReportTypeRow

The ReportTypeRow class contains the information for associating a list of problem sections with a report type. It is specified by the user as part of the TaggingObject JSON string passed in. The ReportTypeRow object contains a string specifying the type of report and a list of Section objects as the sectionsToReport field, which are the problem sections that the user wants a report on. The constructor of the ReportTypeRow requires the report

type and the list of problem sections as parameters. Functions are provided to manipulate the sections in the row, as well as to add new sections or count the number already there.

ItemReportObject

Path: org.assistments.domain.ItemReportObject

The ItemReportObject is one actual implementation of the AbstractBasicReport class, and it represents an actual Item Report. This object contains the information needed to create and display an item report, and can be converted to JSON and sent as part of a response for a report request.

For correctly displaying an Item Report, the ItemReportObject class contains two extra fields in addition to the ones in the AbstractBasicReport. One is a map that maps the problem ID to the correct answers, and the other is a list of ReportObjectColumn named columnList, which will be described later. In order to construct an ItemReportObject object, the list of user IDs, the list of Sections, and the map containing answers of all problems within the sections are needed. And when the constructor is called, the function generateReport() that implements the abstract function in the parent class will be called, so no other action is needed after the constructor of the ItemReportObject is called in order to run the algorithm to process information on students' progress, associate that progress with problems, and generate the report. Furthermore, it is here that class-wide average statistics are calculated. Incidentally, the same procedure should also apply to other report types.

ReportObjectColumn

Path: org.assistments.domain.ReportObjectColumn

The ReportObjectColumn class is created to represent one of the columns in a report, or essentially, the data for one problem or statistic. If there are any header rows, this class can be extended to meet the need, just as the ItemReportColumn class that will be described later.

The ReportObjectColumn contains a map that maps the row headers to the cell contents, which is of type Object for further extension, and a string that represents the header of the column. For constructing a ReportObjectColumn, a string as the header and the map of cell contents are required as parameters. Getters and setters are provided to access and change data for individual cells in the column.

ItemReportColumn

Path: org.assistments.domain.ItemReportColumn

As previously described, the ItemReportColumn is the extension of the ReportObjectColumn that adds three fields as extra rows. These three extra rows are a problem average, common wrong answers, and the correct answer. The constructor of the ItemReportColumn takes all needed arguments in the ReportObjectColumn as well as the value associated with the three extra fields, and the getter functions for the extra fields are also provided.

CompletionReportObject

Path: org.assistments.domain.CompletionReportObject

The CompletionReportObject is another implementation of the AbstractBasicReport class, and it represents an actual Completion Report. By converting it to JSON string, it can be sent as a part of an HTTP response to a report request and used to display a Completion Report.

In addition to the fields of the AbstractBasicReport, the CompletionReportObject also contains a map which this time maps user IDs to a cell representing some aspect of their project on the assignment. In a similar way to the ItemReportObject, the constructor of the CompletionReportObject takes user ID list, the list of sections, the report type, and will call generateReport() function before returns, so that no extra step is needed after calling constructor to run the algorithm to generate the report.

The CompletionReportObject also contains a getReport() function to report the list of columns, and a generateReport() function to process the data and construct the report. In the generateReport() function, the problem logs will be added to the problem section they belong to, and then the cellMap will be updated by iterating through all problem logs to add each student's information as an entry to the map. The Completion Report contains five columns, namely the completion status column, the percentage correct column, the problem indicating each student's problem seen so far, the number of problems done correctly, and the time spent as a whole for each student. In order to construct those five columns, five maps are created, and the information stored in the cellMap are processed to get the five values and added to each map accordingly. Those maps are then added to the column list to serve as the actual columns of the report.

CompletionReportCell

Path: org.assistments.domain.CompletionReportCell

The CompletionReportCell class is created for better helping the creation of the Completion Report. Each CompletionReportCell object is actually a row in the Completion Report, as it could be seen as a map of a student to the student performance data related to each column in the Completion Report. It thus contains the student ID, the number of problems seen by the student, the number of problems done correctly, the total time spent, and information regarding which sections were encountered by the student in the problem set. A CompletionReportCell object must be constructed by giving a student ID. Methods are also provided to update information on each student's statistics, as well as to access and retrieve that information.

Modified files and classes:

AssignmentDao.java

Path: org.assistments.service.dao.jdbc.JdbcAssignmentDao

The function Section getProblemSectionsTree(long assignmentId) is added, which can create a tree of sections to represent a particular assignment.

ClassDao.java

Path: org.assistments.service.dao.ClassDao

The function List<Long> getStudentIdsFromClass(long classId) is added, which can return a list of student IDs from a given class ID.

UserDao.java

Path: org.assistments.service.dao.UserDao

The function List<Long> getUserIdsFromStudentIds(List<Long> studentIds) is added, which converts a list of student IDs to the corresponding user IDs.

Added Dao files

AssistmentToSequenceDao.java

Path: org.assistments.service.dao.AssistmentToSequenceDao

This Dao file contains one function, long getSectionFromAssistment(long assistmentId), which can determine what section a given ASSISTment belongs to.

ProblemDao.java

Path: org.assistments.service.dao.ProblemDao

This Dao file contains one function, void setCorrectAnswers(Map<Long, Answer> answers), that can be used for getting the Answer object from the problem ID. When this function is used, a map should be created and every problem ID that the user wants correct answers from should be added as the key, and the value associated should be an Answer object. By calling this function, the correct answer field in the Answer object will be updated.

Controller

In order to proper handle the incoming HTTP POST request, the file SmartReportController has been created. By following the Spring Framework controller

routine, this class contains a function getSmartReport() to read an HTTP request entity and return a response. The JSON representing a TaggingObject is provided as the HTTP request body, and a reference to a particular assignment is passed as the path variable. Both these fields are needed as parameters of this function.

In this file, multiple Dao objects are used to access the needed resources for further processing. Specifically, all user information, problem logs and correct answers of all problems in the given assignment ID are retrieved, and to further map the relationship between sections and the problem logs, the association between ASSISTment IDs and sections are needed, which is also done by querying the database using Dao objects as described above.

After being able to access all information needed, the section tree within the assignment will be constructed by calling the function getProblemSectionsTree() in the assignmentDao, which returns the root section of the entire tree. Later this root section will be used to traverse the section tree in the way determined by different tagging rules to associate student problem logs with the appropriate sections. Furthermore, these sections can be arranged according to the provided tagging object into the correct type of reports, and thus the list of abstract basic reports can be built to make the smart report.

Based on the content of the tagging JSON string, the decision of whether to follow the given rule or the default rule to generate the report is made. If the JSON String provided is not empty, the TaggingObject converted from the tagging JSON string will be consulted to determine the actual structure of the smart report; however if it is empty, the default algorithm for generating the smart report will be used.

To begin the process of default tagging, we first traverse through the tree of sections and child sections which represents an assignment. At each node in the tree—representing a section—we check to see the type of section it was. If it is a linear section or random section we create an item report from the problems at that node. If, on the other hand, it was a choose condition, we generate a completion report from that data, since generating an item report might give us output similar to that in figure 5. We handle if-then-else conditions similar to choose conditions, except we also report appropriately on the "if" conditional (since that could be a problem set in and of itself). After traversing through the entire tree, in this manner we are able to form a reporting structure which had a section in the final report corresponding to each section in the problem set.

After the section tree is traversed (whether with the default algorithm or by using a TaggingObject), the smart report is complete. The smart report object is then converted into JSON. Then the JSON string is returned as the response to the original HTTP request. The client can then inspect the received JSON string, parse it and utilize the information within to display the smart report.