Find And Assign: An ASSISTments Project in Java

A Major Qualifying Project Report Submitted to the Faculty of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the Degree of Bachelor of Science

by

Christian Roberts

Submitted on
June 1 2017

Approved:

Professor Neil T. Heffernan, Advisor

Cristina Heffernan, Advisor

Table Of Contents

problemSets.jsp
view_custom_problems.jsp
view_problems.jsp
breadcrumb.css
navbar.js

List of Figures

Abstract

ASSISTments is a Learning Management System (LMS) developed and under development at WPI. Its original implementation was in Ruby, but in an effort to integrate with Google Classroom and other Content Management Systems (CMS), development has moved towards a Java implementation. Find and Assign is designed to be put to use by any ASSISTments Java applications in order to get content from ASSISTments and assign them to classes. The assignments are intended for use in any LMS, not just within ASSISTments.

Acknowledgements

I would like to acknowledge my advisors, Neil and Cristina Heffernan, for their guidance and understanding throughout the project. Though at times the work was slow, their understanding and patience allowed for me to produce a useful product that will be central to the new ASSISTments world. They deserve additional acknowledgement in their hands in developing ASSISTments and their passion for the Learning Sciences. I would also like to acknowledge Chris Donnelly and David Magid for the Software Development Kit (SDK) that they made available for ease of development and their frequent management and improvements on said SDK. I would again like to acknowledge Chris Donnelly for being available and willing to answer questions at almost time. Finally, I want to acknowledge the rest of the ASSISTments team; firstly for having many resources available for sharing in the development process as well as for fostering a friendly and professional environment.

Authorship

Christian Roberts wrote this paper in its entirety.

Introduction

ASSISTments is a Learning Management System developed in 2003 intended primarily for use in K-12 learning environments. There were two main objectives in its initial implementation: to deliver free content to students of all ages and to study how well the students learn depending on how the content is presented . This project concerns itself with the prior. ASSISTments was and continues to be a useful tool, but in order to expand its reach it needs to integrate itself more directly with Google Classroom and the other large Learning Management Systems used for grade school and college education. The original implementation of ASSISTments does not lend itself very well to this integration, so development has moved towards creating small Java applications.

The goal of this project was to develop Find and Assign, a Java tool intended for use in the finding certified ASSISTments content and delivering it to the user. The objectives of this project were to do that effectively with Human-Computer Interaction (HCI) design concepts in mind.

Design

To understand the design methodology behind Find and Assign, it needs to be clear what its objective is first. Find and Assign is not intended for use as a stand-alone application, but instead as a tool to be used by other ASSISTments apps. If a user ever requires the functionality that Find and Assign provides, the app they are using will link them to Find and Assign until they have completed their task at which point they will be returned. For this reason, Find and Assign should be minimalistic, not intrusive, and should mimic the style of other ASSISTments applications.

The initial designs did follow this approach, and focused primarily on the customization options, so that the user could feel like the site was designed for them.

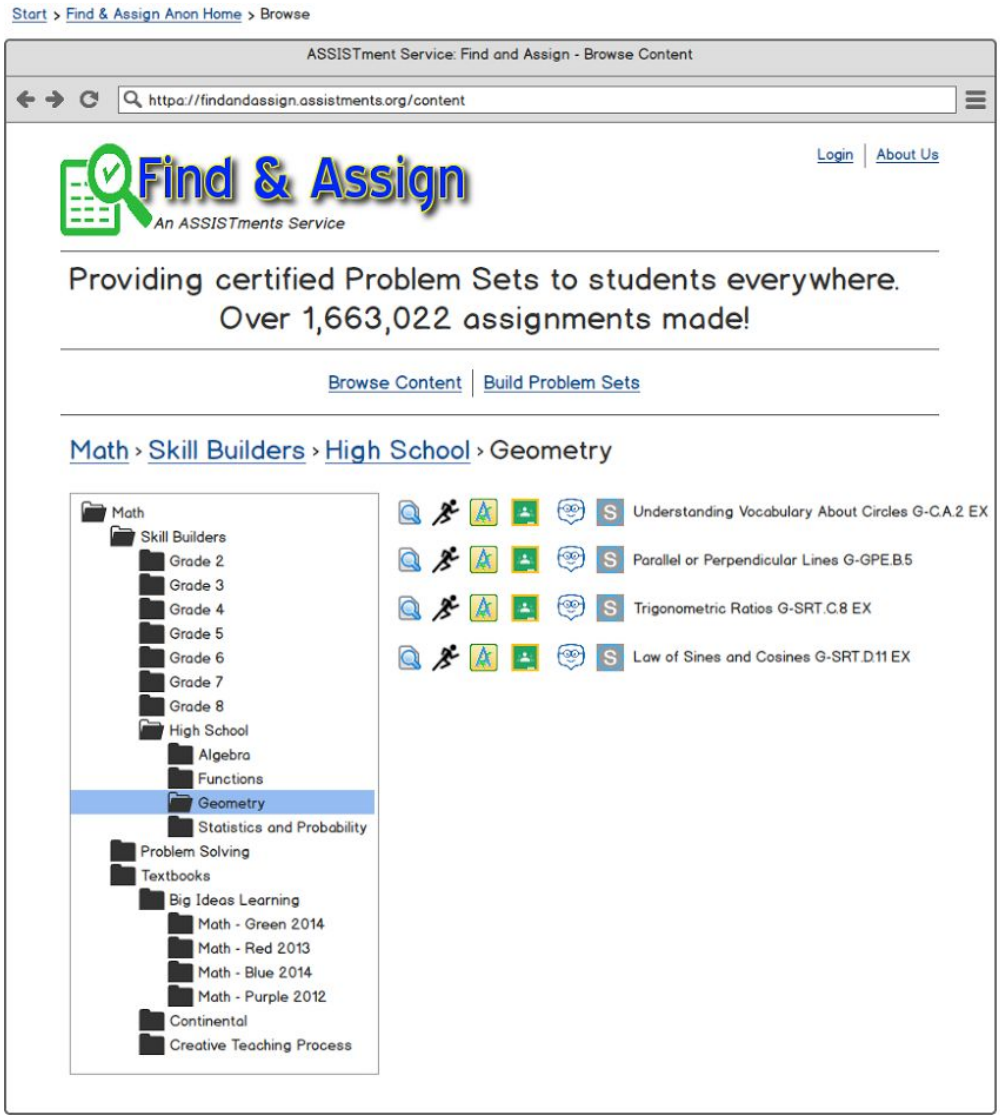


Figure 1: David Magid's initial Find and Assign concept.

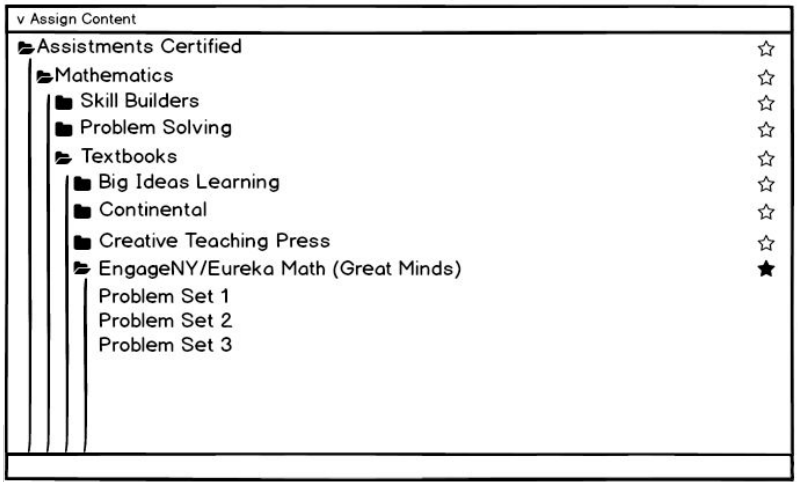


Figure 2: A design experimenting with Default Folders

As Find and Assign was implemented, as is the case with most projects, the design evolved to match a standard set forth by other ASSISTments applications. One ASSISTments tool that had a heavy role in influencing Find and Assign's direction was the ASSISTments App currently available through Google.

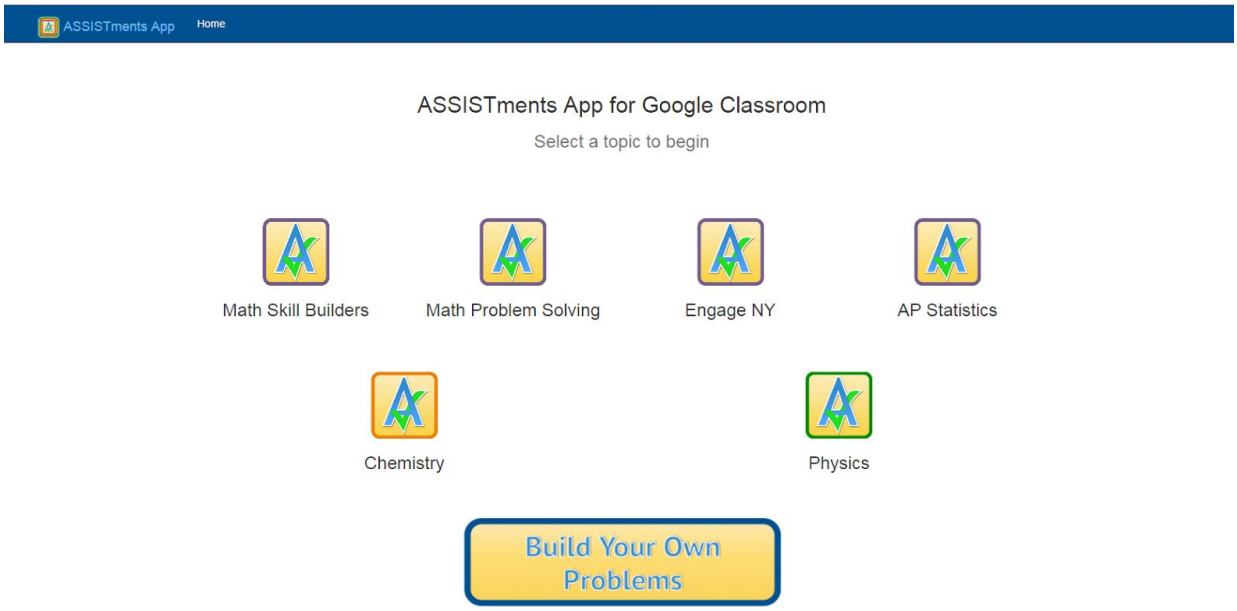


Figure 3: The ASSISTments App

The simplistic style lent itself very well to a streamlined development process, providing a clear objective for a final product. Many of the resources of the different ASSISTments applications were available throughout the process, making it much easier to mimic the look and feel of the page without having to spend too much time on the user interface.

Implementation

Navigation

The first thing you are greeted with when you launch Find and Assign is the folder browser. Be default, there is only one folder present: the ASSISTments Certified folder. This folder contains all of the content that ASSISTments maintains, developed by experts, or taken directly from textbooks with freely available content. Users can assume that content inside ASSISTments Certified is reputable, and that any issues can be reported to ASSISTments and swiftly corrected by the team. Once a user clicks on the folder they wish to navigate to, they will be shown the full list of folders under that one. If a user would like to navigate directly to a known folder, they can save it as a default (this only works if they log in though). Below are images of the process.

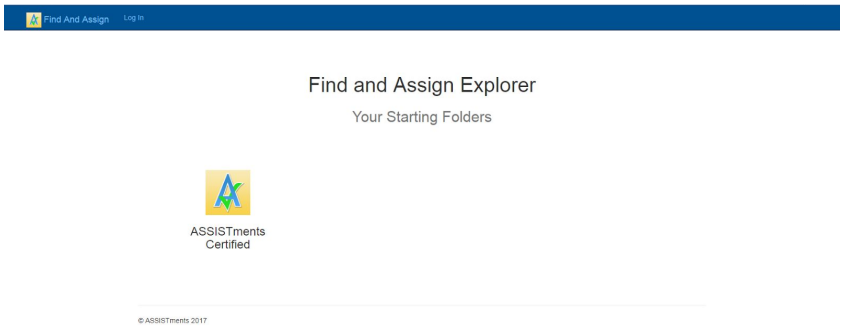


Figure 4: The home page for an unvalidated user.

Figure 5: The navbar for a logged in user.

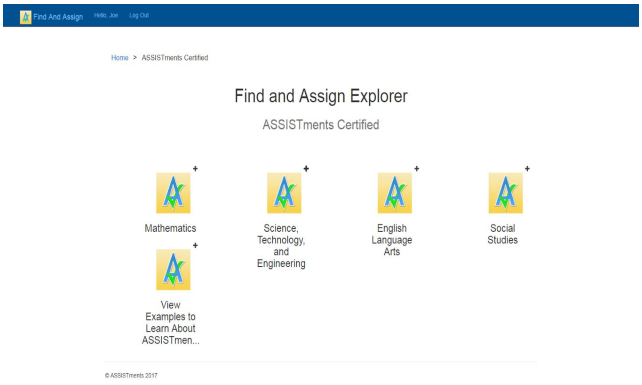


Figure 6: Browsing. The plus buttons add that folder to defaults.

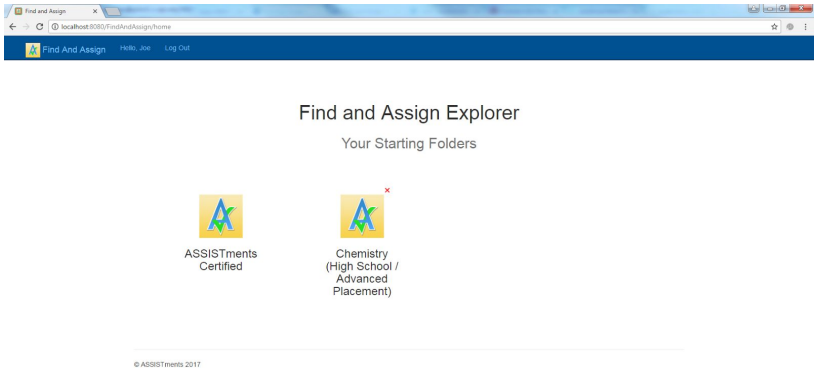
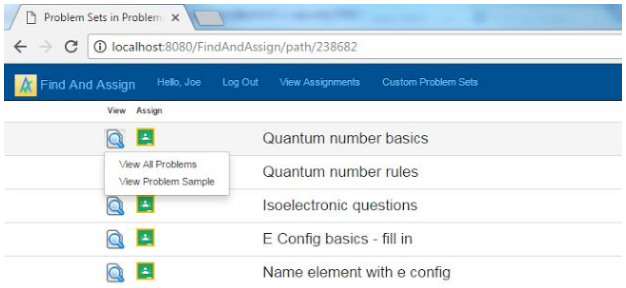


Figure 7: The home page with a default folder.



The user can choose from a list of problem sets once they arrive at the lowest level folder. If they are a verified teacher, they can see all problems within the problem set; otherwise they can only see a problem sample. This is to give anyone the ability to see ASSISTments Certified content samples without allowing students to full problem sets their teachers might be assigning them.

Figure 8: Problem Sets.

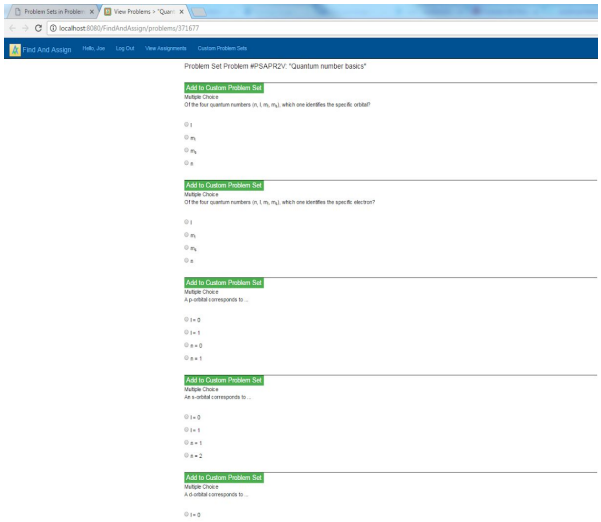


Figure 9: Problems within a Problem Set.

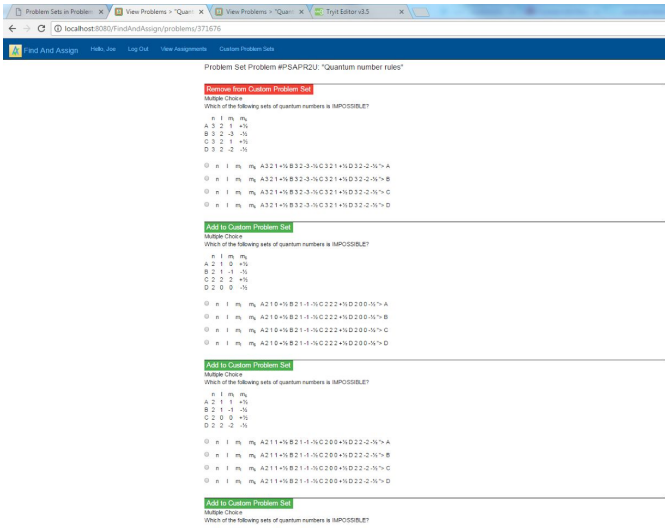


Figure 10: A Problem Set with saved Problems

The Shopping Cart

I wanted to give users the ability to generate their own problem sets when creating assignments. As a result, Find and Assign mimics online shopping models and has a concept of a “Shopping Cart”. The term, though not entirely accurate, is very descriptive of how it works. The shopping cart persists across a session and multiple sessions, such that you may return to Find and Assign and, upon entering with the same credentials, have the same problems available to you. The problems you add can be seen in your custom problem set menu.

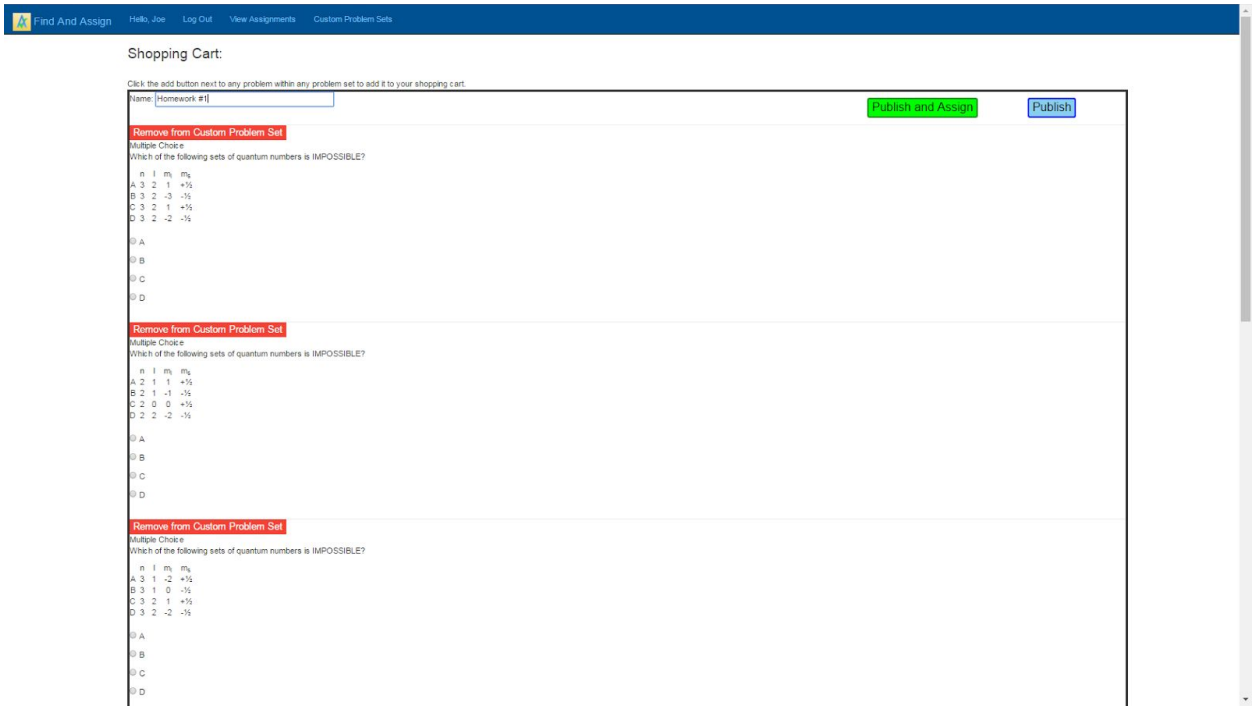


Figure 11: The custom problem set menu with the shopping cart displayed.

The page also displays any other custom problem sets you created through Find and Assign. It lists them below the shopping cart.

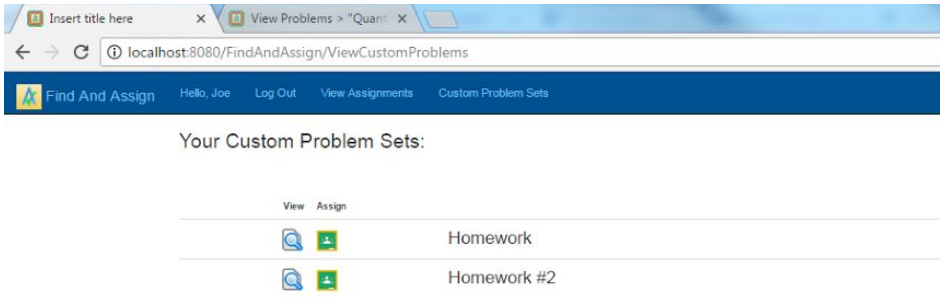


Figure 12: The user’s custom problem sets.

Log-In

Though there were initial designs in creating a login page and process, Chris and David created a login page which could be linked to directly. Once the user has gone through the login page, they are redirected back to Find and Assign along with their credentials. The credentials also include the details necessary in order to determine the user type and the authorities they have (teacher or student, for example).

Figure 13: The “Log In” button

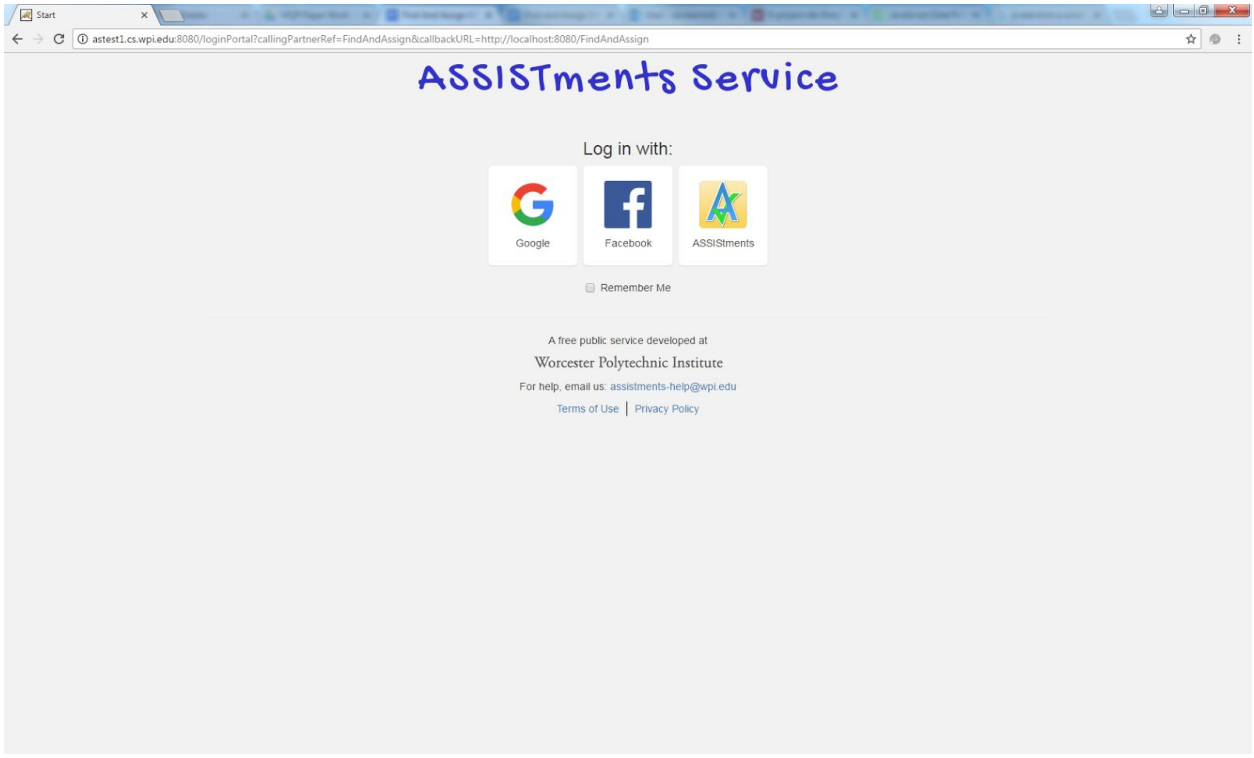


Figure 14: The login screen. A separate service from Find and Assign.

All ASSISTments services will be authenticated through the same login portal. As development goes on for ASSISTments applications, more services will be available to link to in the same fashion as this screen (Find and Assign being one of them). The login site, while functional, has not been through many design iterations, so it is intentionally simplistic to inspire future work.

Assigning

When a user has selected a problem set or created a custom one and they wish to assign it, they can select where they would like to assign it to. Currently, the choices are restricted to a Google Classroom assignment or one only in ASSISTments. The user has to specify where they want to submit the assignment to as the login portal currently fails to determine where the user came from or to which LMS their account is currently linked. Once the user clicks the link, they are shown a dialogue where they can specify the details of their submission.

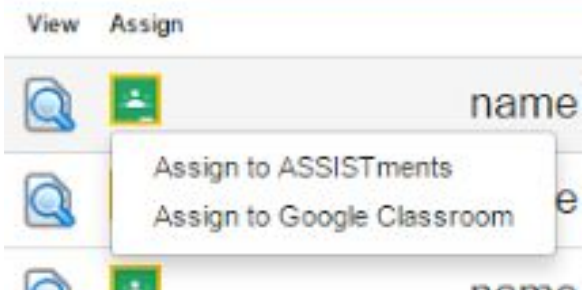


Figure 15: The Assign Button

Figure 16: The Assign Dialogue

The assign dialogue will seamlessly gather all of the user's groups from their respective LMS and list them at the top of the screen. The problem set is then assigned to their LMS group and a link will be provided that will take them to the ASSISTments tutor.

Conclusion and Future Work

Find and Assign is intended as a functional implementation of a very necessary feature in ASSISTments' transition into the Java world. I'm happy with the final product, and it accomplishes effectively what it needs to do. That said, it will not be put to use until its framework is in place. That is to say, whatever other applications will be launching into Find and Assign. Before that happens, there is some work that could still be done in Find and Assign and this section should help illustrate some of my ideas.

Below are features that future developers of Find and Assign can look to implement. These do not have to be features within Find and Assign, but should at least be provided by whatever application the user is coming from. If the design team feels that every user that arrives at Find and Assign will want these features, then Find and Assign can and likely should be the place in which the services are provided.

Viewing Assignments and Reports

A user should be able to see the assignments that were assigned to them or that they have assigned through Find and Assign. Additionally, teachers who have created assignments should be able to generate a report listing every student's status for any given assignment. A student should also be able to see the progress they have made on all of their assignments.

Assignment Dialogue

If a user has come into Find and Assign from a certain LMS, it is necessary that Find and Assign give the feeling that it is integrated with the LMS they are coming from. That is to say that the transition between a user being sent to Find and Assign and returning to their LMS should be consistent to the point that an uneducated user would never get the feeling that they were linked outside of their LMS. The main point in which this can be achieved is in the assignment dialogue that we provide prior to generating the assignment in their LMS. As we support and integrate more LMSs, an assignment dialogue matching their style should be provided. The current one should also be updated to more match the style of Google Classroom, our currently supported LMS.

Automatic LMS Detection

Currently, when a user comes in to find and assign, we do not know which LMS they came from. It is possible, however, to make a call to the LMS and see if they have any groups in existence. When a user logs in to Find and Assign for the first time, we can do a "first time setup" where we individually query each LMS we support and check if our user has groups in that LMS; if they do, we can save that LMS in that user's properties. That way, we can avoid giving users options that are technically unavailable to them (i.e. they should not have an "Assign to Google Classroom" button if they did not come from Google Classroom).

A User Info/Settings Page

A page where a user can see any information pertinent to themselves. Additionally, they could declare permanent settings. I currently have few ideas as to what these settings may be, but as Find and Assign becomes more complex, their nature will become more apparent. One use for them could be to have the user specify their LMSs and their default assignment location. They could also manage all of their properties (the shopping cart, their default folders, and their custom sequences) more closely from this page.

Custom Names for Saved Folders

A saved folder currently use the folder's name as the display name for the user. This can be confusing, as many folder names require context to give additional information. For example, a user can have a folder named "Grade 2" from a Mathematics Folder or from a Science folder (or both). To avoid confusion, custom folders should give the option of allowing the user to define the name.

Complete Folder Structure

The folder browser displays breadcrumbs at the top of the page but the user would have a complete indication of where they are if the complete folder structure were displayed off to the side of the page. Though initial designs toyed with this idea, they were not a priority in implementation. The structure should resemble the structure provided by ASSISTments. The biggest point of contention in implementing this feature is the fear of slowing down the time it takes to browse through files. At the moment, opening a folder is immediate, but having to load up the entire folder structure as you navigate from page to page might augment load times. If that is the case, I would ultimately opt out of implementing this; otherwise I think it would be a great quality of life improvement.

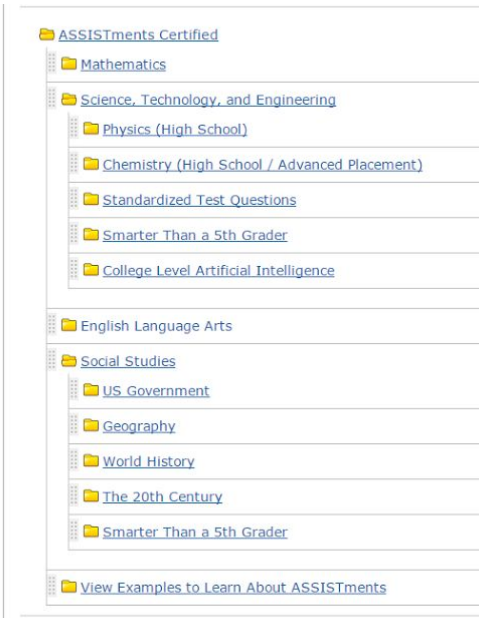


Figure 17: ASSISTments folder structure

Appendix

Code

Managers

CustomProblemManager.java

```java
package org.assistments.findandassign.manager;

import java.time.Instant;
import java.util.LinkedList;
import java.util.List;

import org.assistments.domain.content.Section;
import org.assistments.domain.content.SectionLink;
import org.assistments.domain.content.Sequence;
import org.assistments.service.dao.base.impl.NVPair;
import org.assistments.service.dao.content.SectionDao;
import org.assistments.service.dao.content.SectionLinkDao;
import org.assistments.service.dao.content.SequenceDao;
import org.assistments.service.manager.content.CopyManager;
import org.assistments.service.manager.content.SequenceManager;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class CustomProblemManager {

	@Autowired private CopyManager copyManager;
	@Autowired private SequenceDao sequenceDao;
	@Autowired private SectionDao sectionDao;
	@Autowired private SectionLinkDao sectionLinkDao;
	@Autowired private SequenceManager sequenceManager;
	@Autowired private PropertiesManager propertiesManager;


	/**
	 * Creates a new sequence with a linear head section.
	 * @param sequenceName The Sequence Name.
	 * @param sequenceDescription The description for the new sequence.
	 *
	 * @return The id of the new sequence.
	 */
	private int createSequence(String sequenceName, String sequenceDescription){
		int headSectionId = createSection();
		Sequence newSequence = new Sequence();
		newSequence.setHeadSectionId(headSectionId);
		newSequence.setCreatedAt(Instant.now());
		newSequence.setName(sequenceName);
		newSequence.setDescription(sequenceDescription);
		newSequence.setQualityLevelId(1);
```

```
                newSequence.setUpdatedAt(Instant.now());;
                newSequence.setParametersWithoutParsing("--- {}\n\n");
                int sequenceId = sequenceDao.persist(newSequence);
                propertiesManager.addCustomSequence(sequenceId);
                NVPair pair = new NVPair("sequence_id", sequenceId);
                sectionDao.update(headSectionId, pair);
                return sequenceId;
        }

        private int createSection() {
                Section section = new Section();
                section.setType("LinearSection");
                section.setCreateAt(Instant.now());
                section.setUpdatedAt(Instant.now());
                section.setName("Head Section");
                section.setParametersWithoutParsing("--- {}\n\n");
                return sectionDao.persist(section);
        }

        /**
         * Creates a new sequence with a linear problem sections (creates a problem set with
the given problems).
         * @param problemIdList The list of problem ids to be used in the new problem set.
         * @param name The name of the sequence.
         * @param description The sequence description. Descriptions are usually unused, so
pass an empty string.
         *
         * @return The id for the new sequence/problem set.
         */
        public int createNewSequenceWithProblemIds(List<Integer> problemIdList, String name,
String description){
                List<Integer> astIdList = new LinkedList<Integer>();
                for (Integer problemId : problemIdList){
                        astIdList.add(sequenceManager.findAssistmentIdByProblemId(problemId));
                }
                int sequenceId = createSequence(name, description);
                List<Integer> a =
copyManager.createMultipleProblemSectionsByAstIdList(sequenceId, astIdList);
                createLinks(sequenceManager.findHeadSectionBySequenceId(sequenceId).getId(),
a);
                return sequenceId;
        }

        private void createLinks(int headSectionId, List<Integer> sequenceIds){
                int position = 1;
                for (int sequenceId : sequenceIds){
                        SectionLink link = new SectionLink();
                        link.setParentId(headSectionId);
                        link.setPosition(position);
                        position++;
                        link.setCreateAt(Instant.now());
                        link.setChildId(sequenceId);
                        link.setParameters("--- {}\n\n");
                        link.setUpdatedAt(Instant.now());
                        sectionLinkDao.persist(link);
                }
        }

}
```

FindAndAssignAssignmentManager.java

```
package org.assistments.findandassign.manager;

import java.util.ArrayList;
import java.util.List;

import org.assistments.domain.core.Assignment;
import org.assistments.domain.core.PrincipalType;
import org.assistments.service.dao.base.impl.QueryTerm;
import org.assistments.service.dao.core.AssignmentDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class FindAndAssignAssignmentManager {

	@Autowired AssignmentDao assignmentDao;

	/**
	 * Gets the list of assignments within ASSISTments for a user.
	 * @param userId The id of the user.
	 *
	 * @return The user's assignments.
	 */
	public List<Assignment> getAssignmentsForUser(int userId){
		List<QueryTerm> values = new ArrayList<QueryTerm>();
		values.add(AssignmentDao.Field.ASSIGNEE_XID.getQueryTerm(userId));

values.add(AssignmentDao.Field.ASSIGNEE_PRINCIPAL_TYPE_ID.getQueryTerm(PrincipalType.USER.ge
tId()));
		return assignmentDao.findAllObjects(values);
	}
}
```

FolderManager.java

```
package org.assistments.findandassign.manager;

import java.util.ArrayList;
import java.util.List;
import org.assistments.domain.legacy.LegacyFolder;
import org.assistments.domain.legacy.LegacyFolderItem;
import org.assistments.domain.legacy.CertifiedFolder;
import org.assistments.service.dao.base.impl.QueryTerm;
import org.assistments.service.dao.legacy.LegacyCertifiedFolderDao;
import org.assistments.service.dao.legacy.LegacyFolderDao;
import org.assistments.service.dao.legacy.LegacyFolderItemDao;
import org.assistments.service.exceptions.NotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
```

```java
@Component
public class FolderManager{

	@Autowired private LegacyFolderItemDao folderItemDao;
	@Autowired private LegacyFolderDao folderDao;
	@Autowired private LegacyCertifiedFolderDao certifiedFolderDao;
	/**
	 * Finds the Legacy Folder with the given id.
	 * @param id The ID.
	 *
	 * @return The legacy folder.
	 * @throws NotFoundException
	 */
	public LegacyFolder findFolderById(int id) throws NotFoundException {
		List<QueryTerm> values = new ArrayList<QueryTerm>();
		values.add(LegacyFolderDao.Field.ID.getQueryTerm(id));
		return folderDao.findObject(values);
	}

	/**
	 * Finds the subfolders of a given legacy folder. If null, the folder id belongs to a
leaf folder
	 * and you might be looking for findAllProblemSetsByFolderId().
	 * @param id The folder ID.
	 *
	 * @return A list of all Legacy Folders with the given folder ID as their Parent_ID.
	 * @throws NotFoundException
	 */
	public List<LegacyFolder> findSubfoldersById(int id) throws NotFoundException {
		List<QueryTerm> values = new ArrayList<QueryTerm>();
		values.add(LegacyFolderDao.Field.PARENT_ID.getQueryTerm(id));
		System.out.println("Checking now!");
		try {
			return folderDao.findAllObjects(values);
		} catch (NotFoundException e){
			return null;
		}
	}

	/**
	 * Gets the LegacyFolderItem of the given type. Legacy Folder Items have a position
field, so useful
	 * for sorting folders.
	 * @param id The item ID.
	 * @param type The type of item. One of {"CurriculumItem", "DataFile", "Folder",
"ClassAssignment"}
	 *
	 * @return The LegacyFolderItem meeting the requirements.
	 * @throws NotFoundException
	 */
	public LegacyFolderItem findItemById(int id, String type) throws NotFoundException {
		List<QueryTerm> values = new ArrayList<QueryTerm>();
		values.add(LegacyFolderItemDao.Field.ITEM_ID.getQueryTerm(id));
		values.add(LegacyFolderItemDao.Field.ITEM_TYPE.getQueryTerm(type));
		return folderItemDao.findAllObjects(values).get(0);
	}

	/**
	 * Recursively gets the CertifiedFolders representing the problem sets with the given
ID as the parent folder.
```

```
         * @param folderId The parent folder's ID.
         *
         * @return A list of CertifiedFolders with type "CurriculumItem".
         */
        public List<CertifiedFolder> findAllProblemSetsByFolderId(int folderId){
                List<CertifiedFolder> tempList =
certifiedFolderDao.findAllFolderById(folderId);
                List<CertifiedFolder> psList = new ArrayList<CertifiedFolder>();

                for(CertifiedFolder cf : tempList){
                        if(cf.getType().equals("CurriculumItem")){
                                psList.add(cf);
                        }else{
                                psList.addAll(findAllProblemSetsByFolderId(cf.getFolderId()));
                        }
                }

                return psList;
        }
}
```

ProblemManager.java

```
package org.assistments.findandassign.manager;

import static java.lang.Math.toIntExact;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;

import org.assistments.domain.content.AnswerRow;
import org.assistments.domain.content.ProblemRow;
import org.assistments.domain.content.tutor.Manifest;
import org.assistments.service.dao.base.impl.QueryTerm;
import org.assistments.service.dao.content.ProblemDao;
import org.assistments.service.dao.content.SequenceDao;
import org.assistments.service.dao.legacy.LegacyCurriculumItemDao;
import org.assistments.service.manager.content.ManifestManager;
import org.assistments.service.manager.content.PersistableKeyManager;
import org.assistments.service.manager.content.SequenceManager;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class ProblemManager {

        @Autowired private SequenceManager sequenceManager;
        @Autowired private SequenceDao sequenceDao;
        @Autowired private LegacyCurriculumItemDao curriculumDao;
        @Autowired private ManifestManager manifestManager;
        @Autowired private ProblemDao problemDao;

        /**
```

```
        * Gets ASSISTments problems for the given problem set ID.
        * @param problemSetId The problem set ID.
        * @param count The number of unique problems to be acquired.
        *
        * @return A map representing the problems, each entry value is a map representing
the problem.
        * Each problem has the following fields: {"problemText", "problemType", "id",
"answers"}
        * Answers is a Map where every entry value represent a possible answer for the
problem.
        * Each answer has the following fields: {"answerText", "isCorrect"}
        */
       public Map<String, Object> findAssistmentsProblems(int problemSetId, int count) {
              Random rand = new Random(522207140);
              Map<String, Object> requestedObject = new HashMap<String, Object>();
              List<ProblemRow> problemList = new ArrayList<ProblemRow>();
              try {
                     int headSectionId =
sequenceManager.findHeadSectionBySequenceId(problemSetId).getId();
                     List<Integer> astIdList =
sequenceManager.findAllAssistmentsIdsByHeadSectionId(headSectionId);
                     String astIdListDtr = "";
                     for(int astId : astIdList){
                            astIdListDtr += "," + String.valueOf(astId);
                            ProblemRow temp =
sequenceManager.findFirstMainProblemByAssistmentId(astId);
                            problemList.add(temp);
                     }
                     astIdListDtr = astIdListDtr.substring(1);
              }catch(Exception e){
                     requestedObject.put("1", "empty");
                     return requestedObject;
              }
              int i = 0;
              while (i < count || count == -1) {
                     if (problemList.size() == 0) break;

                     int idx = 0;
                     if (count != -1){
                            idx = rand.nextInt(problemList.size());
                     }
                     ProblemRow problem = problemList.get(idx);
                     problemList.remove(idx);

                     String problemType =
getProblemTypeByTypeId(problem.getRubyProblemType().getId());

                     List<AnswerRow> answerList = new ArrayList<AnswerRow>();
                     try{
                            answerList =
sequenceManager.findAllAnswersByProblemId(problem.getId());
                     }catch(Exception e){
                            //e.printStackTrace();
                            continue;
                     }
                     if (answerList.size() < 1) continue;

                     List<Map<String, Object>> answerArray = new ArrayList<Map<String,
Object>>();
                     for(AnswerRow a : answerList){
```

```
                        Map<String, Object> answerMap = new HashMap<String, Object>();
                        answerMap.put("answerText", a.getValue());
                        answerMap.put("isCorrect", a.getIsCorrect());
                        answerArray.add(answerMap);
                }

                Map<String, Object> currentProblem = new HashMap<String, Object>();

                currentProblem.put("id", problem.getId());
                currentProblem.put("problemSetId", problemSetId);
                currentProblem.put("problemText", problem.getBody());
                currentProblem.put("answers", answerArray);
                currentProblem.put("problemType", problemType);
                currentProblem.put("creatorUserId", 0);
                currentProblem.put("randomizeAnswers", problem.getRandomizeAnswers());

                i++;
                requestedObject.put(Integer.toString(i), currentProblem);
        }

        if (requestedObject.isEmpty()) {
                System.out.println("Or here.");
                requestedObject.put("1", "empty");
        };
        return requestedObject;
}

/**
 * Gets the name of the given problem set.
 * @param problemSetId The Problem Set/Sequence ID.
 *
 * @return The Problem Set Name.
 */
public String getProblemSetName(int problemSetId){
        return sequenceManager.findSequenceBySequenceId(problemSetId).getName();
}

/**
 * Gets a ProblemRow object from the problem ID.
 * @param problemId The Problem ID.
 *
 * @return A ProblemRow object with Problem ID = problemId.
 */
public ProblemRow getProblemByProblemId(int problemId){
        List<QueryTerm> values = new ArrayList<QueryTerm>();
        values.add(ProblemDao.Field.ID.getQueryTerm(problemId));
        return problemDao.findAllObjects(values).get(0);
}

/**
 * Converts a problem type id to a string.
 * @param id The problem type ID (within the range 1-15).
 *
 * @return The problem type as a string. An empty string if the ID is invalid.
 */
public String getProblemTypeByTypeId(int id) {
        switch(toIntExact(id)){
        case 1:
                return "Multiple Choice";
        case 2:
```

```
                        return "Check All That Apply";
                case 3:
                        return "Rank";
                case 4:
                        return "Exact Match (case sensitive)";
                case 5:
                        return "Algebraic Expression";
                case 8:
                        return "Ungraded Open Response";
                case 10:
                        return "Write A Racket Problem";
                case 11:
                        return "Exact Match (ignore case)";
                case 12:
                        return "Externally Run";
                case 13:
                        return "Number";
                case 14:
                        return "Numeric Expression";
                case 15:
                        return "Exact Fraction";
                default:
                        return "null";
                }
        }

        public Map<String, Object> findAllAssistmentsProblems(int problemId) {
                return findAssistmentsProblems(problemId, -1);
        }
}
```

PropertiesManager.java

```
package org.assistments.findandassign.manager;

import java.util.LinkedList;
import java.util.List;

import org.assistments.domain.core.properties.ObjectProperty;
import org.assistments.domain.core.properties.ObjectPropertyTargetType;
import org.assistments.service.exceptions.NotFoundException;
import org.assistments.service.manager.core.ObjectPropertyManager;
import org.assistments.service.security.authentication.core.AuthenticationHolder;
import org.assistments.util.Util;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class PropertiesManager {
        @Autowired ObjectPropertyManager propertyManager;

        /**
         * Gets the signed in users default folders.
         *
         * @return A list of folderIDs that are the user's DefaultFolders property.
         */
```

```
	public List<Integer> getDefaultFolders(){
		try{
			return toList((String)
propertyManager.getValue(AuthenticationHolder.getCurrentUser(),
"DefaultFolders",ObjectPropertyTargetType.USER).getValue());
		} catch(NotFoundException e){
			return null;
		}
	}

	/**
	 * Gets the user's "Shopping Cart".
	 *
	 * @return The list of Problem IDs.
	 */
	public List<Integer> getShoppingCart(){
		try{
			return toList((String)
propertyManager.getValue(AuthenticationHolder.getCurrentUser(),
"ShoppingCart",ObjectPropertyTargetType.USER).getValue());
		} catch(NotFoundException e){
			return null;
		}
	}

	/**
	 * Gets the Custom Sequences the user has created through Find and Assign.
	 *
	 * @return A list of Custom Sequence IDs.
	 */
	public List<Integer> getCustomSequences(){
		try{
			return toList((String)
propertyManager.getValue(AuthenticationHolder.getCurrentUser(),"CustomSequences",
ObjectPropertyTargetType.USER).getValue());
		} catch(NotFoundException e){
			return null;
		}
	}

	private void setIdList(String propName, int id){
		List<Integer> values = null;
		try {
			String vString = (String)
propertyManager.getValue(AuthenticationHolder.getCurrentUser(), propName,
ObjectPropertyTargetType.USER).getValue();
			values = toList(vString);
		} catch (NotFoundException e){
			values = new LinkedList<Integer>();
		}
		values.add(id);
		values = removeDupes(values);
		propertyManager.setValue(AuthenticationHolder.getCurrentUser(),
ObjectPropertyTargetType.USER, propName, toString(values));
	}

	private void removeElementFromIdList(String propName, int id){
		try {
			ObjectProperty prop =
propertyManager.getValue(AuthenticationHolder.getCurrentUser(), propName,
```

```
ObjectPropertyTargetType.USER);
                        List<Integer> list = toList((String) prop.getValue());
                        list.remove(new Integer(id));
                        propertyManager.setValue(AuthenticationHolder.getCurrentUser(),
ObjectPropertyTargetType.USER, propName, toString(list));
                } } catch(NotFoundException e){

                }
        }

        /**
         * Removes a custom sequence from the user's object property.
         * @param sequenceId The custom sequence ID to be removed.
         */
        public void removeCustomSequence(int sequenceId){
                removeElementFromIdList("CustomSequences", sequenceId);
        }

        /**
         * Checks if a folder ID is one of the default folders. Not recommended to check
several folders at once, in that case it's best to compare
         * directly to the list.
         * @param folderId The ID to check.
         *
         * @return True if folderId is default. False otherwise.
         */
        public boolean isDefaultFolder(int folderId){
                return this.getDefaultFolders().contains(folderId);
        }

        /**
         * Removes the given default folder from the user's DefaultFolders property.
         * @param folderId The folder ID.
         */
        public void removeDefaultFolder(int folderId){
                removeElementFromIdList("DefaultFolders", folderId);
        }

        /**
         * Removes the given problem from the user's ShoppingCart property.
         * @param folderId The problem ID.
         */
        public void removeFromShoppingCart(int problemId){
                removeElementFromIdList("ShoppingCart", problemId);
        }

        /**
         * Adds the given default folder to the user's DefaultFolders property.
         * @param folderId The folder ID.
         */
        public void setDefaultFolder(int folderId){
                setIdList("DefaultFolders", folderId);
        }

        /**
         * Adds the given problem to the user's ShoppingCart property.
         * @param folderId The problem ID.
         */
        public void addToShoppingCart(int problemId){
                setIdList("ShoppingCart", problemId);
```

```
	}

	/**
	 * Adds the given problem to the user's ShoppingCart property.
	 * @param folderId The problem ID.
	 */
	public void addCustomSequence(int sequenceId){
		setIdList("CustomSequences", sequenceId);
	}

	/**
	 * Deletes the shopping cart.
	 */
	public void clearShoppingCart(){
		propertyManager.deleteValue(AuthenticationHolder.getCurrentUser(),
ObjectPropertyTargetType.USER, "ShoppingCart");
	}

	private List<Integer> toList(String vString) {
		if (Util.isNullOrEmpty(vString)) return new LinkedList<Integer>();
		String[] parts = vString.split(",");
		List<Integer> list = new LinkedList<Integer>();
		for (String term : parts){
			list.add(Integer.parseInt(term));
		}
		return list;
	}

	private String toString(List<Integer> v){
		if (v.isEmpty()) return "";
		String output = "" + v.get(0);
		for (int i = 1; i < v.size(); i++){
			output += "," + v.get(i);
		}
		return output;
	}

	private List<Integer> removeDupes(List<Integer> list){
		List<Integer> result = new LinkedList<Integer>();
		for (Integer i : list){
			if (!result.contains(i)){
				result.add(i);
			}
		}
		return result;
	}
}
```

Test

TestController.java

```
package org.assistments.findandassign.test;

import java.util.ArrayList;
import java.util.LinkedList;
```

```
import java.util.List;
import java.util.Map;

import org.assistments.domain.content.ProblemRow;
import org.assistments.domain.core.Assignment;
import org.assistments.domain.core.properties.ObjectPropertyDefinition;
import org.assistments.findandassign.manager.FolderManager;
import org.assistments.findandassign.manager.ProblemManager;
import org.assistments.findandassign.manager.PropertiesManager;
import org.assistments.findandassign.manager.CustomProblemManager;
import org.assistments.findandassign.manager.FindAndAssignAssignmentManager;
import org.assistments.service.manager.content.SequenceManager;
import org.assistments.service.manager.core.GroupManager;
import org.assistments.service.manager.core.ObjectPropertyManager;
import org.assistments.service.security.authentication.core.AuthenticationHolder;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class TestController {

	@Autowired FolderManager manager;
	@Autowired ProblemManager problemManager;
	@Autowired GroupManager groupManager;
	@Autowired CustomProblemManager	customProblemManager;
	@Autowired SequenceManager sequenceManager;
	@Autowired PropertiesManager propertiesManager;
	@Autowired ObjectPropertyManager propertyManager;
	@Autowired FindAndAssignAssignmentManager assignmentManager;

	@RequestMapping(value = "/addGroup", method = RequestMethod.GET)
	public String addGroup(){
		groupManager.persist("Test Group", AuthenticationHolder.getCurrentUser());
		return "redirect:home";
	}

	@RequestMapping(value = "/testAllThings", method=RequestMethod.GET)
	public String doThis(){
		System.out.println(manager.findAllProblemSetsByFolderId(1).size());
		return "redirect:home";
	}

	@RequestMapping(value = "/createAndAssignTest", method=RequestMethod.GET)
	public String createAndAssignTest(){
		List<Integer> problemIds = new LinkedList<Integer>();
		problemIds.add(1089003);
		problemIds.add(1088902);
		problemIds.add(1089003);
		int sequenceId =
customProblemManager.createNewSequenceWithProblemIds(problemIds, "Test Sequence", "");
		System.out.println("Sequence Created");
		System.out.println(sequenceId);
		int headSectionId =
sequenceManager.findHeadSectionBySequenceId(sequenceId).getId();
		System.out.println("Head Section Id: " + headSectionId);
		List<Integer> astIdList =
sequenceManager.findAllAssistmentsIdsByHeadSectionId(headSectionId);
```

```
                System.out.println("Stuff gotten.");

                List<ProblemRow> problemList = new ArrayList<ProblemRow>();
                for(int astId : astIdList){
                        ProblemRow temp =
sequenceManager.findFirstMainProblemByAssistmentId(astId);
                        problemList.add(temp);
                }
                for (ProblemRow p : problemList){
                        System.out.println(p.getBody());
                }
                return "redirect:customProblem/" + sequenceId;
        }

        @RequestMapping(value="/testProperties", method=RequestMethod.GET)
        public String testProperties(@RequestParam("folder") int folderId){
                propertiesManager.setDefaultFolder(folderId);
                return "redirect:home";
        }

        @RequestMapping(value="/printProperties", method=RequestMethod.GET)
        public String printProperties(){
                for (ObjectPropertyDefinition a :propertyManager.getMyPropertyDefinitions()){
                        System.out.println(a.getObjectProperty().getName() + ":" +
a.getObjectProperty().getValue());
                }
                return "index";
        }

        @RequestMapping(value="/testPropertiesRemove", method=RequestMethod.GET)
        public String testPropertiesRemove(@RequestParam("folder") int folderId){
                propertiesManager.removeDefaultFolder(folderId);
                return "redirect:home";
        }

        @RequestMapping(value="/assignmentsTest", method=RequestMethod.GET)
        public String assignmentsTest(){
                for (Assignment a :
assignmentManager.getAssignmentsForUser(AuthenticationHolder.getCurrentUser().getXid())){
                        System.out.println(a.getName());
                }
                return "index";
        }
}
```

Web

AssignmentsController.java

```
package org.assistments.findandassign.web;

import java.time.Instant;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.Optional;
```

```java
import javax.servlet.http.HttpServletRequest;

import org.assistments.domain.core.Assignment;
import org.assistments.domain.core.ExternalReferenceType;
import org.assistments.domain.core.LmsProviderType;
import org.assistments.domain.core.User;
import org.assistments.domain.external.XInteractionType;
import org.assistments.service.domain.lms.LmsGroupProfile;
import org.assistments.service.exceptions.NotFoundException;
import org.assistments.service.manager.builders.AssignmentBuilder;
import org.assistments.service.manager.builders.XInteractionBuilder;
import org.assistments.service.manager.core.AssignmentManager;
import org.assistments.service.manager.core.GroupManager;
import org.assistments.service.manager.external.XisManager;
import org.assistments.service.manager.lms.LmsForbiddenException;
import org.assistments.service.manager.lms.LmsIntegrationManager;
import org.assistments.service.manager.lms.LmsManagerProvider;
import org.assistments.service.manager.lms.impl.LmsRequestAttributes;
import org.assistments.service.security.authentication.core.AuthenticationHolder;
import org.assistments.service.security.authorization.NeedsLmsAuthentication;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class AssignmentsController {

	@Autowired private XisManager interactionMgr = null;
	@Autowired private AssignmentManager assignmentManager;
	@Autowired private LmsManagerProvider lmsMgrProvider;
	@Autowired private GroupManager groupManager;


	@RequestMapping(value = "/assignments", method=RequestMethod.GET)
	String getAssignments(){
		return "assignments";
	}

	@NeedsLmsAuthentication
	@RequestMapping(value = "/assignToStream", method= RequestMethod.GET)
	public String assignTestToStream(
			@RequestParam(LmsRequestAttributes.LMS_PROVIDER_TYPE) int lmsPtype,
			@RequestParam("Name") String name,
			@RequestParam("Description") Optional<String> description,
			@RequestParam("DueDate") Date dueDate,
			@RequestParam("sequenceId") Integer problemSetId,
			@RequestParam("Group") String groupXRef,
			//@RequestParam("isUser") boolean isUser,
			//@RequestParam("gradeId") Integer gradeId,
```

```
                        HttpServletRequest request){

                Assignment a = assignHelper(problemSetId, dueDate , name, groupXRef);
                System.out.println("Successfully assignmed my stuff: "+a.getName());


                if(!pushAssignmentsToStream(lmsPtype, a, description.isPresent()?
description.get(): "")){
                        System.out.println("ERROR");
                };
                //map.put("test_name", test.getName());
                //map.put("test_xref", test.getAssignment().getXref());
                //map.put("test_xid", test.getAssignment().getXid());*/


                return "redirect:home";

        }


        @RequestMapping(value = "/assignToAssistments", method= RequestMethod.GET)
        public String assignToAssistments(
                        @RequestParam(LmsRequestAttributes.LMS_PROVIDER_TYPE) int lmsPtype,
                        @RequestParam("Name") String name,
                        @RequestParam("Description") Optional<String> description,
                        @RequestParam("DueDate") Optional<Date> dueDate,
                        @RequestParam("sequenceId") Integer problemSetId,
                        @RequestParam("Group") String groupXRef,
                        HttpServletRequest request){

                Assignment a = assignHelper(problemSetId, dueDate.isPresent()? dueDate.get():
new Date(), name, groupXRef);
                System.out.println("Successfully assignmed my stuff: "+a.getName());

                /*
                if(!pushAssignmentsToStream(map, lmsPtype, "The test instructions go here!
FIXME", "The remediation instructions go here! FIXME", ab)){
                        map.put("error_message", "AssignmnentNotCreated!!!");
                        return new ResponseEntity<Map<String, Object>>(map,
HttpStatus.BAD_REQUEST);
                }; */
                //map.put("test_name", test.getName());
                //map.put("test_xref", test.getAssignment().getXref());
                //map.put("test_xid", test.getAssignment().getXid());*/


                return "redirect:home";

        }

        private Assignment assignHelper(int problemSetId, Date dueDate, String name, String
groupXRef){
                User currentUser = AuthenticationHolder.getCurrentUser();
                AssignmentBuilder ab = new AssignmentBuilder(problemSetId, currentUser);
                ab.setDueDate(dueDate.toInstant());
                ab.setReleaseDate(Instant.now());

ab.assignToGroup(groupManager.getGroup(ExternalReferenceType.GROUP.getXInfo(groupXRef)));
                ab.setAssignmentName(name);
                return assignmentManager.persist(ab);
```

```
        }

        private boolean pushAssignmentsToStream(int lmsPtype,  Assignment a, String
description) {

                LmsIntegrationManager mgr =
this.lmsMgrProvider.getManager(LmsProviderType.findById(lmsPtype));

                new HashMap<String,Object>();
                XInteractionBuilder ib = new
XInteractionBuilder(XInteractionType.ASSIGNMENT_URL);


            ib.setAssignmentXref(a.getXref());        //Should be assignmentID?

            ib.setVersion(XisManager.CURRENT_ASSIGNMENT_URL_VERSION);

            //extraInfo.put("onExit", "home"); //These lines are for changing the return
address.

            //ib.setExtraInfo(extraInfo);


      try {

            String intId = interactionMgr.persistUrl(ib) ;
                  //FIXME change url to final url.
       mgr.streamAssignmentXiid(intId, "localhost:8080", LmsProviderType.findById(lmsPtype),
description); //RunAnywhere host should be here


      } catch (Exception e) {
         e.printStackTrace();
         return false;
      }

      return true;
       }


        @ModelAttribute()
        private void setDefaultModel(Model model){
                ControllerHelper.setDefaultModel(model);
        }
}
```

ControllerHelper.java

```
package org.assistments.findandassign.web;

import org.assistments.domain.core.User;
import org.assistments.service.security.authentication.core.AuthenticationData;
import org.assistments.service.security.authentication.core.AuthenticationHolder;
import org.assistments.service.security.authorization.GlobalRoleType;
import org.springframework.ui.Model;

public class ControllerHelper {
```

```
	public static void setDefaultModel(Model model){
		AuthenticationData data = AuthenticationHolder.getAuthentication();
		if (data.isAuthenticated()){
			model.addAttribute("isUser", true);
			User loggedInUser = data.getPrincipal();
			model.addAttribute("userFirstName", loggedInUser.getName());
			model.addAttribute("userLastName", loggedInUser.getLastName());
			model.addAttribute("userDisplayName", loggedInUser.getDisplayName());
			if
(data.getAuthorities().contains(GlobalRoleType.TEACHER.getGrantedAuthority())){
				model.addAttribute("isTeacher", true);
			} else{
				model.addAttribute("isTeacher", false);
			}
		} else{
			model.addAttribute("isUser", false);
		}
	}
}
```

CreateAssignmentController.java

```
package org.assistments.findandassign.web;

import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import org.assistments.domain.core.Group;
import org.assistments.domain.core.Group.GroupChildDepth;
import org.assistments.domain.core.LmsProviderType;
import org.assistments.findandassign.manager.PropertiesManager;
import org.assistments.service.domain.lms.LmsGroupProfile;
import org.assistments.service.exceptions.AlreadyExistsException;
import org.assistments.service.exceptions.NotFoundException;
import org.assistments.service.manager.content.SequenceManager;
import org.assistments.service.manager.core.GroupManager;
import org.assistments.service.manager.lms.LmsForbiddenException;
import org.assistments.service.manager.lms.LmsIntegrationManager;
import org.assistments.service.manager.lms.LmsManagerProvider;
import org.assistments.service.manager.lms.impl.LmsRequestAttributes;
import org.assistments.service.security.authentication.core.AuthenticationHolder;
import org.assistments.service.security.authorization.NeedsLmsAuthentication;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class CreateAssignmentController {
```

```
	@Autowired private GroupManager groupManager;
	@Autowired private LmsManagerProvider lmsMgrProvider;
	@Autowired private PropertiesManager propertiesManager;
	@Autowired private SequenceManager sequenceManager;

	@RequestMapping(value = "/assign/{problemSetId}", method= RequestMethod.GET)
	public String assign(Model model, @PathVariable("problemSetId") int
sequenceId,@ModelAttribute("isTeacher") boolean isTeacher){
		if (!isTeacher){
			System.out.println("Only teachers are allowed here!");
			return "redirect/login";
		}
		model.addAttribute("assignButtonText", "Assign");
		return createAssignmentHelper(model, sequenceId);
	}

	@NeedsLmsAuthentication
	@RequestMapping(value = "/assignToLms/{problemSetId}", method = RequestMethod.GET)
	public String importGroups(Model model,
			@PathVariable("problemSetId") int sequenceId,
			@RequestParam(LmsRequestAttributes.LMS_PROVIDER_TYPE) int lmsPtype,
			@ModelAttribute("isTeacher") boolean isTeacher){
		if (!isTeacher){
			System.out.println("Only teachers are allowed here!");
			return "redirect/login";
		}
		LmsIntegrationManager lmsIntegrationManager =
lmsMgrProvider.getManager(LmsProviderType.findById(lmsPtype));
		try {
			for (LmsGroupProfile group: lmsIntegrationManager.getLmsGroups()){
				try{
					System.out.println(group.getUid());
					System.out.println(group.getLmsType());
					lmsIntegrationManager.importLmsGroup(group, true);
				} catch(AlreadyExistsException e){
					System.out.println("Group exists!");
					continue;
				}
			}
		} catch (NotFoundException | LmsForbiddenException e) {
			System.out.println("User must have not come from an LMS.");
		}
		model.addAttribute("assignButtonText", "Assign to " +
LmsProviderType.findById(lmsPtype).getName());
		model.addAttribute("lmsPtype", lmsPtype);
		return createAssignmentHelper(model, sequenceId);
	}

	private String createAssignmentHelper(Model model, int sequenceId){

model.addAttribute("Custom",propertiesManager.getCustomSequences().contains(sequenceId));
		List<Map<String, Object>> groupsInfo = new LinkedList<Map<String, Object>>();
		for (Group group
:groupManager.getOwnedGroups(AuthenticationHolder.getCurrentUser())){
			Map<String, Object> map = new HashMap<String, Object>();
			map.put("name", group.getName());
			map.put("memberCount",
group.getMemberCount(GroupChildDepth.ANY_CHILD));
			map.put("xRef", group.getXref());
			groupsInfo.add(map);
```

```
                }
                model.addAttribute("Groups", groupsInfo);
                model.addAttribute("ProblemSetId", sequenceId);
                model.addAttribute("ProblemSetName",
sequenceManager.findSequenceBySequenceId(sequenceId).getName());
                return "assignTo";
        }

        @ModelAttribute()
        private void setDefaultModel(Model model){
                ControllerHelper.setDefaultModel(model);
        }
}
```

CustomProblemController.java

```
package org.assistments.findandassign.web;

import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Optional;

import javax.servlet.http.HttpServletRequest;

import org.assistments.domain.content.AnswerRow;
import org.assistments.domain.content.ProblemRow;
import org.assistments.domain.content.Sequence;
import org.assistments.findandassign.manager.CustomProblemManager;
import org.assistments.findandassign.manager.ProblemManager;
import org.assistments.findandassign.manager.PropertiesManager;
import org.assistments.service.manager.content.SequenceManager;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class CustomProblemController {

        @Autowired private PropertiesManager propertiesManager;
        @Autowired private SequenceManager sequenceManager;
        @Autowired private ProblemManager problemManager;
        @Autowired private CustomProblemManager customProblemManager;

        @RequestMapping("/ViewCustomProblems")
        public String viewCustomProblems(Model model) {
                List<String> sequenceNames = new LinkedList<String>(), sequenceDescriptions =
new LinkedList<String>();
                List<Integer> sequenceIds = propertiesManager.getCustomSequences();
                for (int sequenceId: sequenceIds){
                        Sequence sequence =
sequenceManager.findSequenceBySequenceId(sequenceId);
                        sequenceNames.add(sequence.getName());
                        sequenceDescriptions.add(sequence.getDescription());
```

```
                }
                model.addAttribute("CustomSetIds", sequenceIds);
                model.addAttribute("CustomSetNames", sequenceNames);
                model.addAttribute("CustomSetDescriptions", sequenceDescriptions);
                Map<String, Map<String, Object>> problems = new HashMap<String, Map<String,
Object>>();
                int cur = 0;
                for (int problemId: propertiesManager.getShoppingCart()){
                        Map<String, Object> map = new HashMap<String, Object>();
                        ProblemRow problem = problemManager.getProblemByProblemId(problemId);
                        map.put("problemText", problem.getBody());
                        map.put("id", problem.getId());
                        map.put("problemType",
problemManager.getProblemTypeByTypeId(problem.getRubyProblemType().getId()));
                        map.put("inCart", true);
                        List<AnswerRow> answers =
sequenceManager.findAllAnswersByProblemId(problem.getId());
                        List<Map<String, Object>> answerMapList = new LinkedList<Map<String,
Object>>();
                        for (AnswerRow answer: answers){
                                Map<String, Object> a = new HashMap<String, Object>();
                                a.put("answerText", answer.getValue());
                                a.put("isCorrect", answer.getIsCorrect());
                                answerMapList.add(a);
                        }
                        System.out.println(answers);
                        map.put("answers", answerMapList);
                        problems.put(Integer.toString(cur), map);
                        cur++;
                }
                model.addAttribute("problemMap", problems);
                return "view_custom_problems";
        }

        @RequestMapping("/publishSet")
        public String publishNewCustomSet(HttpServletRequest request,
                        @RequestParam("Name") String name,
                        @RequestParam("Description") Optional<String> description){
                String desc;
                if (description.isPresent()){
                        desc = description.get();
                } else{
                        desc = "";
                }
                int sequenceId =
customProblemManager.createNewSequenceWithProblemIds(propertiesManager.getShoppingCart(),
name, desc);
                propertiesManager.addCustomSequence(sequenceId);
                propertiesManager.clearShoppingCart();
                String referer = request.getHeader("Referer");
            return "redirect:"+ referer;
        }

        @ModelAttribute()
        private void setDefaultModel(Model model){
                ControllerHelper.setDefaultModel(model);
        }
}
```

FolderExplorerController.java

```
package org.assistments.findandassign.web;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.assistments.domain.core.AssistmentsPrincipal;
import org.assistments.domain.core.Group;
import org.assistments.domain.legacy.CertifiedFolder;
import org.assistments.domain.legacy.LegacyFolder;
import org.assistments.findandassign.manager.FolderManager;
import org.assistments.findandassign.manager.PropertiesManager;
import org.assistments.service.exceptions.NotFoundException;
import org.assistments.service.manager.core.GroupManager;
import org.assistments.service.security.authentication.core.AuthenticationHolder;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class FolderExplorerController {

	@Autowired private FolderManager folderManager;
	@Autowired private GroupManager     groupManager;
	@Autowired private PropertiesManager propertiesManager;

	@RequestMapping(value = "/path/{folderId}" , method = RequestMethod.GET)
	public String goToFile(@PathVariable("folderId") int folderId, Model model)
	{
		LegacyFolder temp = null;
		List<LegacyFolder> temp2 = null;
		boolean problems = false;
		try {
			temp = folderManager.findFolderById(folderId);
			temp2 = folderManager.findSubfoldersById(folderId);
			if (temp2 == null){//Looking at problem sets.
				if
(AuthenticationHolder.getAuthentication().isAuthenticated()){
					try{
						List<Group> groups =
groupManager.getOwnedGroups(AuthenticationHolder.getCurrentUser());
						System.out.println("Do you have groups? " +
groups.size());
						for (Group g : groups){
							System.out.println(g.getName());

System.out.println(g.getType().toString());
							for (AssistmentsPrincipal a :
g.getMembers(Group.GroupChildDepth.ANY_CHILD)){
								System.out.println("This is one of
```

```
the children:");
                                                        System.out.println(a.getName());
                                                }
                                        }
                                } catch(NotFoundException e){
                                        e.printStackTrace();
                                }
                        }
                        problems = true;
                        List<CertifiedFolder> cfs =
folderManager.findAllProblemSetsByFolderId(folderId);
                        List<String> childrenNames = new ArrayList<String>();
                        List<Integer> childrenIDs = new ArrayList<Integer>();
                        for (CertifiedFolder cf : cfs){
                                childrenNames.add(cf.getItemName());
                                childrenIDs.add(cf.getProblemSetId());
                        }
                        model.addAttribute("ProblemSetNames", childrenNames);
                        model.addAttribute("ProblemSetIds", childrenIDs);
                } else{//Looking at folders.
                        Collections.sort(temp2, (o1, o2) ->
folderManager.findItemById(o2.getId(), "Folder").getPosition() -
folderManager.findItemById(o1.getId(), "Folder").getPosition());
                        List<String> childrenNames = new ArrayList<String>();
                        List<Integer> childrenIDs = new ArrayList<Integer>();
                        List<Boolean> isDefault = new ArrayList<Boolean>();
                        List<Integer> defaults = new ArrayList<Integer>();
                        if (AuthenticationHolder.getAuthentication().isAuthenticated())
defaults = propertiesManager.getDefaultFolders();
                        for (LegacyFolder temp1 : temp2){
                                childrenNames.add(temp1.getName());
                                childrenIDs.add(temp1.getId());
                                isDefault.add(defaults.contains(temp1.getId()));
                        }
                        model.addAttribute("folderNames", childrenNames);
                        model.addAttribute("folderIDs", childrenIDs);
                        if (AuthenticationHolder.getAuthentication().isAuthenticated())
model.addAttribute("isDefaultFolder", isDefault);
                }
        } catch (NotFoundException e) {
                // This should not happen
                Map<String, Object> error = new HashMap<String, Object>();
                error.put("error", "ASSISTments has encountered an error. "
                                + "Please send an email to assistments-help@wpi.edu"
                                + "with problem set name.");
                //return new ResponseEntity<Map<String, Object>>(error,
HttpStatus.OK);
        }
        model.addAttribute("parentName", temp.getName());
        model.addAttribute("parentID", temp.getId());
        List<String> pathNames = new ArrayList<String>();
        List<Integer> pathIDs = new ArrayList<Integer>();
        if (temp.getParentId() != 0){
                temp = folderManager.findFolderById(temp.getParentId());
        }
        while (temp.getParentId() != 0){
                pathNames.add(0, temp.getName());
                pathIDs.add(0, temp.getId());
                temp = folderManager.findFolderById(temp.getParentId());
        }
```

```
                model.addAttribute("pathNames", pathNames);
                model.addAttribute("pathIDs", pathIDs);
                if (problems){
                        return "problemSets";
                }
                return "folderBrowser";
        }

        @ModelAttribute()
        private void setDefaultModel(Model model){
                ControllerHelper.setDefaultModel(model);
        }

}
```

IndexController.java

```
package org.assistments.findandassign.web;


import java.util.LinkedList;
import java.util.List;

import org.assistments.domain.legacy.LegacyFolder;
import org.assistments.findandassign.manager.FolderManager;
import org.assistments.findandassign.manager.PropertiesManager;
import org.assistments.service.manager.core.RolesManager;
import org.assistments.service.security.authentication.core.AuthenticationData;
import org.assistments.service.security.authentication.core.AuthenticationHolder;
import org.assistments.service.security.authorization.GlobalRoleType;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;


@Controller
public class IndexController {

        @Autowired private FolderManager findAndAssignFolderManager;
        @Autowired private RolesManager rolesManager;
        @Autowired private PropertiesManager propertiesManager;

        @RequestMapping(value = {"/home", "/", ""} , method = RequestMethod.GET)
        public String home(Model model) {
                AuthenticationData data = AuthenticationHolder.getAuthentication();
                if (data.isAuthenticated()){
                        LegacyFolder temp = null;
                        List<Integer> folderIdList = propertiesManager.getDefaultFolders();
                        System.out.println(folderIdList);
                        if (folderIdList != null){
                                List<String> names = new LinkedList<String>();
                                for (Integer folderId : folderIdList){
                                        temp =
findAndAssignFolderManager.findFolderById(folderId);
                                        names.add(temp.getName());
```

```
                        }
                        model.addAttribute("folderIds", folderIdList);
                        model.addAttribute("folderNames", names);
                }

                }
                return "index";
        }

        //FIXME: delete me. I am for testing
        @RequestMapping("/addTeacherRole")
        public void addTeacherRole()
        {
                rolesManager.addUserRole(AuthenticationHolder.getCurrentUser(),
GlobalRoleType.TEACHER.getGrantedAuthority());
        }

        @ModelAttribute()
        private void setDefaultModel(Model model){
                ControllerHelper.setDefaultModel(model);
        }
}
```

LoginController.java

```
package org.assistments.findandassign.web;

import java.util.Optional;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.assistments.domain.core.User;
import org.assistments.service.security.authentication.core.AuthenticationData;
import org.assistments.service.security.authentication.core.AuthenticationHolder;
import org.assistments.service.security.authentication.core.webapp.AccountControllerHelper;
import
org.assistments.service.security.authentication.core.webapp.DefaultAccountControllerHelperIm
pl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class LoginController {

        @Autowired
        DefaultAccountControllerHelperImpl logoutObject;

        @Autowired
        AccountControllerHelper helper;

    private @Value("${login.portal.url.root}") String loginPortalUrlRoot;
```

```
	@RequestMapping(value = "/login" , method = RequestMethod.GET)
	public String logIn(@RequestParam("callbackURL") Optional<String> url, Model model){
		AuthenticationData data = AuthenticationHolder.getAuthentication();
		if (data.isAuthenticated()){
			User loggedInUser = data.getPrincipal();
			System.out.println("Authenticated!");
			System.out.println(loggedInUser.getDisplayName() + ": " +
loggedInUser.getFirstName() + " " + loggedInUser.getLastName());
			if (url.isPresent()) return "redirect:" + url.get();
			return "redirect:home";
		} else {
			System.out.println("Not authenticated!");
			String url2 = loginPortalUrlRoot +
"?callingPartnerRef=FindAndAssign&callbackURL=";
			if (url.isPresent()){
				url2 += url.get()+"/";
				System.out.println(url.get());
			} else{
				url2 += "http://localhost:8080/FindAndAssign/home";
			}
			return "redirect:" + url2;
		}
	}

	  @RequestMapping(method = RequestMethod.GET, value="/logout")
	  public String logout(HttpServletRequest request,HttpServletResponse response)
	  {
	    helper.signOff(request, response);
		return "redirect:" + loginPortalUrlRoot +
"/signout?callingPartnerRef=FindAndAssign&callbackURL=http://localhost:8080/FindAndAssign/ho
me";
	  }
}
```

ProblemController.java

```
package org.assistments.findandassign.web;

import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Optional;

import javax.servlet.http.HttpServletRequest;

import org.assistments.domain.content.AnswerRow;
import org.assistments.domain.content.ProblemRow;
import org.assistments.domain.content.tutor.ManifestCode;
import org.assistments.domain.content.tutor.ProblemSet;
import org.assistments.domain.core.ExternalReferenceType;
import org.assistments.findandassign.manager.ProblemManager;
import org.assistments.findandassign.manager.PropertiesManager;
import org.assistments.service.manager.content.SequenceManager;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
```

```
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class ProblemController {

	@Autowired private SequenceManager sequenceManager;
	@Autowired private ProblemManager problemManager;
	@Autowired private PropertiesManager propertiesManager;

	@RequestMapping(value="/problems/{ProblemSetId}", method=RequestMethod.GET)
	public String showProblems(@PathVariable("ProblemSetId") int problemId,
@RequestParam("Special") Optional<Boolean> studentView,
			Model model,
			HttpServletRequest request,
			@ModelAttribute("isTeacher") boolean isTeacher){
		ProblemSet ps = new
ProblemSet(ManifestCode.PREVIEW_PROBLEM_SET.getShortCode() + problemId);
		ps.setName("Problem #" +
ExternalReferenceType.PROBLEM_SET.getXref(problemId));
		model.addAttribute("problemSetName",
problemManager.getProblemSetName(problemId));
		model.addAttribute("problemSet", ps);
		Map<String, Object> plist = problemManager.findAssistmentsProblems(problemId,
10);
		if (isTeacher && !studentView.isPresent()){
			plist = problemManager.findAllAssistmentsProblems(problemId);
			System.out.println(plist);
			List<Integer> cart = propertiesManager.getShoppingCart();
			for (Entry<String, Object> problem: plist.entrySet()){
				@SuppressWarnings("unchecked")
				Map<String, Object> map = (Map<String, Object>)
problem.getValue();
				map.put("inCart", cart.contains(map.get("id")));
			}
		}
		model.addAttribute("problemMap", plist);
		return "view_problems";
	}

	@RequestMapping(value="/customProblem/{SequenceId}", method=RequestMethod.GET)
	public String getBySequence(@PathVariable("SequenceId") int sequenceId, Model model,
@ModelAttribute("isTeacher") boolean isTeacher){
		model.addAttribute("custom", true);
		List<Integer> astIdList =
sequenceManager.findAllAssistmentsIdsByHeadSectionId(sequenceManager.findHeadSectionBySequen
ceId(sequenceId).getId());
		model.addAttribute("problemSetName",
sequenceManager.findSequenceBySequenceId(sequenceId).getName());
		Map<String, Object> map = new HashMap<String, Object>();
		int num = 0;
		List<Integer> cart = new LinkedList<Integer>();
		if (isTeacher){
			cart = propertiesManager.getShoppingCart();
		}
		for (Integer astId : astIdList){
			ProblemRow problem =
```

```
sequenceManager.findFirstMainProblemByAssistmentId(astId);
                        Map<String, Object> problemMap = new HashMap<String, Object>();
                        problemMap.put("problemText", problem.getBody());
                        problemMap.put("problemType",
problemManager.getProblemTypeByTypeId(problem.getRubyProblemType().getId()));
                        problemMap.put("id", problem.getId());
                        if (isTeacher) problemMap.put("inCart",
cart.contains(problem.getId()));
                        List<AnswerRow> answers =
sequenceManager.findAllAnswersByProblemId(problem.getId());
                        List<Map<String, Object>> answerMapList = new LinkedList<Map<String,
Object>>();
                        for (AnswerRow answer: answers){
                                Map<String, Object> a = new HashMap<String, Object>();
                                a.put("answerText", answer.getValue());
                                a.put("isCorrect", answer.getIsCorrect());
                                answerMapList.add(a);
                        }
                        problemMap.put("answers", answerMapList);
                        map.put(Integer.toString(num), problemMap);
                        num++;
                }
                model.addAttribute("problemMap", map);
                return "view_problems";
        }

        @ModelAttribute()
        private void setDefaultModel(Model model){
                ControllerHelper.setDefaultModel(model);
        }
}
```

PropertiesController.java

```
package org.assistments.findandassign.web;

import javax.servlet.http.HttpServletRequest;

import org.assistments.findandassign.manager.PropertiesManager;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class PropertiesController {
        @Autowired private PropertiesManager propertiesManager;

        @RequestMapping("/removeDefault/{folderId}")
        public String removeDefault(@PathVariable("folderId") int folderId,
HttpServletRequest request){
                propertiesManager.removeDefaultFolder(folderId);
                String referer = request.getHeader("Referer");
            return "redirect:"+ referer;
        }

        @RequestMapping("/addDefault/{folderId}")
        public String addDefault(@PathVariable("folderId") int folderId, HttpServletRequest
```

```
request){
                propertiesManager.setDefaultFolder(folderId);
                String referer = request.getHeader("Referer");
            return "redirect:"+ referer;
        }

        @RequestMapping(value="/addToCart/{ProblemId}")
        public String addToCart(@PathVariable("ProblemId") int problemId, HttpServletRequest
request){
                propertiesManager.addToShoppingCart(problemId);
                String referer = request.getHeader("Referer");
            return "redirect:"+ referer;
        }

        @RequestMapping(value="/removeFromCart/{ProblemId}")
        public String removeFromCart(@PathVariable("ProblemId") int problemId,
HttpServletRequest request){
                propertiesManager.removeFromShoppingCart(problemId);
                String referer = request.getHeader("Referer");
            return "redirect:"+ referer;
        }
}
```

Pages

assignTo.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html5>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Assign To Which Group?</title>
</head>
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/bootstrap.min.css" />" />
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/bootstrap-theme.min.css" />" />
<link rel="shortcut icon"
  href="<c:url value="/resources/images/shortcut_logo.png" />" />
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/jquery-ui.css" />" />
<script
        src="<c:url value="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"
/>"></script>
<script
        type = "text/javascript" src="<c:url value="/resources/js/bootstrap.min.js"
/>"></script>
<script type="text/javascript"
                src="<c:url value="/resources/js/navbar.js" />"></script>
<link rel="shortcut icon" href="images/shortcut_logo.png">
<link rel="stylesheet" href="stylesheets/sharelinks.css">
<!-- Bootstrap core CSS -->
<link rel="stylesheet"
```

```
href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-awesome.min.css">
<!-- Optional theme -->
<link href="stylesheets/bootstrap-theme.min.css" rel="stylesheet">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script type="text/javascript" src="js/bootstrap.min.js"></script>
<script type="text/javascript" src="js/bootstrap-tour.min.js"></script>
<script src="https://apis.google.com/js/platform.js" async defer></script>
<script src="https://apis.google.com/js/client.js"></script>
<link href="stylesheets/bootstrap-select.min.css" rel="stylesheet">
<script type="text/javascript"

src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-select/1.7.5/js/bootstrap-select.min.j
s"></script>
<script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.14.1/moment-with-locales.min.js"></s
cript>
<script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-datetimepicker/4.17.37/js/bootstrap-da
tetimepicker.min.js"></script>
<link
href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-datetimepicker/4.17.37/css/bootstrap-
datetimepicker.min.css" rel="stylesheet">
<script>
	$(function(){
		$("#assign_button").click(function(){
				var name = $("#assignment_title").val();
				var description = $("#assignment_description").val();
				var dueDate = $("#due_date").val();
				var lmsPtype = $("#lms_ptype").text();
				var problemSetId = $("#problem_set_id").text();
				alert("Running");
				var groupId =
$("#courses_select").find("option:selected").attr("id");
				alert(groupId);
				var url = "";
				if (lmsPtype.length == 0){
					alert("Why am I here?");
					url += "/FindAndAssign/assignToAssistments?";
				} else{
					alert("I should be here");
					url += "/FindAndAssign/assignToStream?lmsPtype=" +
lmsPtype +"&";
				}
				alert(lmsPtype);
				url  += "Name=" + name;
				if (description != null){
					url += "&Description=" + description;
				}
				url += "&DueDate=" + dueDate;
				url += "&sequenceId=" + problemSetId;
				url += "&Group="+groupId;
				//url += "&isUser=false";
				window.location.assign(url);

		});

	});
</script>
<body>
```

```
                <span id="lms_ptype" hidden="true">${lmsPtype}</span>
                <span id="problem_set_id" hidden="true">${ProblemSetId}</span>
                                        <button type="button" class="close" data-dismiss="modal"
                                                aria-label="Close">
                                                <span aria-hidden="true">&times;</span>
                                        </button>
                                        <h4 class="modal-title" id="myModalLabel">Assigning
<c:if test="${Custom}">Custom </c:if>Problem Set: ${ProblemSetName}</h4>
                                <form id='assign_form'>
                                        <div class="form-group" style="margin: 0 0 20px 0;">
                                        <b>Choose class</b> <select id="courses_select"
class="selectpicker"  data-width="100%" required>
                                        <c:forEach items="${Groups}" var="group">
                                                <option value="${group.name}"
id="${group.xRef}">${group.name}</option>
                                        </c:forEach>
                                        </select>
                                        </div>

                                                <div class="form-group">
                                                <b>Title </b><input type="text"
id="assignment_title" class="form-control" value="${ProblemSetName}" required>
                                                </div>
                                                <br>
                                                <div class="form-group">
                                                <b>Due Date  <i></i></b><br>
                                                <div class="container">
                                                    <div class="row">
                                                        <div class='col-sm-6'>
                                                            <div class="form-group">
                                                                <div class='input-group date'
id='due_datetime'>

                                                                    <input type='text'

class="form-control"  id="due_date"/>

                                                                    <span

class="input-group-addon">

                                                                        <span class="glyphicon

glyphicon-calendar"></span>

                                                                    </span>
                                                                </div>
                                                            </div>
                                                        </div>
                                                        <script type="text/javascript">
                                                            $(function () {
                                                               var now = new Date();
                                                               //var now_utc = new
Date(now.getUTCFullYear(), now.getUTCMonth(), now.getUTCDate(),  now.getUTCHours(),
now.getUTCMinutes(), now.getUTCSeconds());
                                                               now.setHours(now.getHours()+1)

$('#due_datetime').datetimepicker({

                                                                     "format": "MM/DD/YYYY

HH:mm"

                                                                });

$('#due_datetime').data("DateTimePicker").minDate(now);
                                                            });
                                                        </script>
                                                    </div>
                                                </div>
```

```
                                        </div>
                                        <div class="form-group">
                                               <b>Description <i>(optional)</i></b>
                                               <textarea class="form-control" rows="3"
id="assignment_description"></textarea>
                                        </div>
                                </form>
                                <div class="alert alert-success" role="alert"
id="created_success">
                                       The assignment is successfully created in your Google
Classroom. <br>
                                </div>
                                       <button type="button" class="btn btn-primary"
id="assign_button">${assignButtonText}</button>
                                       <div id='loading_icon' hidden = "true"><i class="fa
fa-refresh fa-spin fa-lg fa-fw"></i><span class="sr-only">Loading...</span>     </div>
                                        <a href="https://classroom.google.com" class="btn
btn-primary"
                                               role="button" id="open_classroom"
target="_blank" onclick="$('#myModal').modal('hide') ">
                                               Open Google Classroom</a>

</body>
</html>
```

folderBrowser.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
       pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Find and Assign</title>
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/bootstrap.min.css" />" />
  <link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/breadcrumb.css" />" />
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/bootstrap-theme.min.css" />" />
<link rel="shortcut icon"
  href="<c:url value="/resources/images/shortcut_logo.png" />" />
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/jquery-ui.css" />" />
<script
       src="<c:url value="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"
/>"></script>
<script
       type = "text/javascript" src="<c:url value="/resources/js/bootstrap.min.js"
/>"></script>
<script
       src="<c:url value="https://apis.google.com/resources/js/platform.js" />"></script>
<script
       src="<c:url value="https://apis.google.com/resources/js/client.js" />"></script>
<style type="text/css">
body {
```

```
        padding-top: 50px;
        padding-bottom: 20px;
}
.apps_link:hover{
        cursor:pointer;
}
.apps_link2:hover{
        cursor:pointer;
}
.container_customize{
        padding:0 60px 0 30px;
}
a:hover{
        cursor:pointer;
}

.problem_building_img :hover {
        cursor:pointer;
}
.nav_active{
        color:white !important;
}
.footer_section{
        position:fixed;
        bottom:10px;
        text-align:center;
}
.remove_default:hover{
        cursor:pointer;
}
.remove_default {
        position:absolute;
        left:145px;
        bottom:calc(100% - 35px);
}
.add_default:hover{
        cursor:pointer;
}
.add_default {
        position:absolute;
        left:145px;
        bottom:calc(100% - 35px);
}

.folder_name {
  word-wrap:normal;
  display:block;
  -o-text-overflow:ellipsis;
  text-overflow:ellipsis;
  width: 150px;
  text-align:center;
}

.apps_link {
        text-align: center;
        text-overflow:ellipsis;
        -o-text-overflow:ellipsis;
        height:150px;
}
.apps_link2 {
```

```
	text-align: center;
}

</style>
<script>
var CLASSROOM_CLIENT_ID =
"757036402283-8o3nu8pdve8snhj8ds11te8bnsrnmuu6.apps.googleusercontent.com";
var SCOPES = ["https://www.googleapis.com/auth/classroom.profile.emails",
              "https://www.googleapis.com/auth/classroom.rosters.readonly",
              "https://www.googleapis.com/auth/classroom.courses"];
var toolSelected = "";
	$(function(){
		$(".apps_link").click(function(){
			var id = $(this).attr("id");
			toolSelected = id;
			locationStr = "${requestScope.location }";
			locationStr = encodeURIComponent(locationStr);
			if(id == "assistments_certified"){
				window.location.assign("/FindAndAssign/path/1"+locationStr);
			}else if (id == "engageNY"){
				window.location.assign("/FindAndAssign/path/254673"
					+locationStr);
			}else if (id == "chemistry_link"){

window.location.assign("/direct/SkillBuilderGoogleClassroom?folder_id=226695&tool_type=chemi
stry&location="
					+locationStr);
			} else if(id == "problem_solving_link") {

window.location.assign("/direct/SkillBuilderGoogleClassroom?folder_id=231370&tool_type=probl
em_solving&location="+
					locationStr);
			}else if(id == "login_button"){
				window.location.assign("/FindAndAssign/login");
			}else{
				window.location.assign("/FindAndAssign/path/" + id);
			}
		});

		$(".remove_default").click(function(){
			var id = $(this).attr("id");
			window.location.assign("/FindAndAssign/removeDefault/" + id);
		});

		$(".add_default").click(function(){
			var id = $(this).attr("id");
			window.location.assign("/FindAndAssign/addDefault/" + id);
		})

		$(".apps_link2").click(function(){
			var id = $(this).attr("id");
			var title = $(this).attr("title");
			<%session.setAttribute("ProblemSetTitle", "${title}");%>
			window.location.assign("/FindAndAssign/problems/" + id);
		});
		$(".folder_name").each(function(){
			var showChar = 30;
			var content = $(this).html();
			if(content.length > showChar){
				var html = "<h3>" + content.substr(0, showChar) + '...' +
```

```
"</h3>";
                                var theRest = content.substr(showChar);
                                $(this).html(html);
                        }
                });
                $("breadcrumb").click(function(){
                        var id = $(this).attr("id");
                        window.location.assign("/FindAndAssign/path/" + id);
                });
                $(".apps_link").hover(function(){
                        $("img",this).animate({
                                width:"+=6"
                        },100);
                },function(){
                        $("img",this).animate({
                                width:"-=6"
                        },100);
                });
                $(".remove_default").hover(function(){
                        $("img",this).animate({
                                width:"+=2"
                        },100);
                },function(){
                        $("img",this).animate({
                                width:"-=2"
                        },100);
                });
                $(".add_default").hover(function(){
                        $("img",this).animate({
                                width:"+=2"
                        },100);
                },function(){
                        $("img",this).animate({
                                width:"-=2"
                        },100);
                });
                $(".problem_building_img").hover(function() {
                        $("img",this).animate({
                                width:"+=8",
                                height: "+=8"
                        },100);
                },function(){
                        $("img",this).animate({
                                width:"-=8",
                                height: "-=8"
                        },100);
                });
        });

        function imageClick() {
                gapi.auth.authorize({
                         'client_id': CLASSROOM_CLIENT_ID,
                         'scope': SCOPES,
                         'immediate': true}, handleAuthResult);
        }

        function handleAuthResult(authResult) {
                if (authResult && !authResult.error) {
                        var firstName;
                        var lastName;
```

```
                        var emailAddr;
                        var ownerId;
                        //get user profile
                    var request = gapi.client.request({
                                root : 'https://classroom.googleapis.com',
                                path : 'v1/userProfiles/me',
                        });

                        request.execute(function(resp) {
                                if(resp.error != null) {
                                if(resp.error.code == 403) {
                                        $("#message_body").html("Sorry... Google Classroom is
not available for your google account at this time.");
                                        $("#message_body").css("color", "blue");
                                        $('#message_modal').modal('show');
                                        return;
                                }
                                $("#message_body").html("Sorry... Something goes wrong here
with Google Classroom. But we have no idea what causes the problem.");
                                $("#message_body").css("color", "red");
                                $('#message_modal').modal('show');
                                return;
                        }

                                firstName = resp.name.givenName;
                                lastName = resp.name.familyName;
                                emailAddr = resp.emailAddress;
                                ownerId = resp.id;
                                $.ajax({
                                        url: 'google_classroom/go_to_assistments',
                                        type: 'POST',
                                        data: {google_user_id: ownerId, first_name: firstName,
last_name: lastName, email_address: emailAddr},
                                        async: false,
                                        success: function(data) {
                                                window.location.replace(data.location);
                                        },
                                        error: function(data) {
                                                $("#message_body").html("Sorry... Something goes
wrong here. We cannot direct you to assistments.org."+
                                                                "Please email us at
assistments-help@wpi.edu with this error.");
                                        $("#message_body").css("color", "blue");
                                        $('#message_modal').modal('show');
                                        }
                                });
                        });
                } else {
                        $("#message_body").html("It seems that you don't log into Google.
Please sign into Google first and then access the app.");
                $("#message_body").css("color", "blue");
                $('#message_modal').modal('show');
                }
        }
</script>
</head>
<body>
        <jsp:include page="navbar.jsp"/>

        <!-- Main jumbotron for a primary marketing message or call to action -->
```

```
	<div class="jumbotron" style="background-color:white;">
		<div class="container">
			<ul class="breadcrumb" style="background-color:white">
			<li> <a href="/FindAndAssign/home">Home</a></li>
			<c:if test = "${parentID != 1 }">
				<li id = 1> <a href="1">
					ASSISTments Certified
				</a></li>
			</c:if>
			<c:forEach items = "${pathIDs}" var = "pathId" varStatus = "status">
				<li id="${pathId}"> <a href="${pathId}">
					${pathNames.get(status.index)}
				</a></li>
			</c:forEach>
				<li>${parentName}</li>
			</ul>
			<div style="text-align: center;">
			<h1><small style="color:#333;">Find and Assign Explorer</small></h1>
			<h2 style="color:#777;">${parentName}</h2>
			</div>
			<div class="row" style="margin-top:70px;">
			<c:forEach items = "${folderIDs}" var = "folderID" varStatus =
"status">
				<div class="col-md-2 col-md-offset-1">
					<!-- FIXME If folder is not in default folders... Hoo
boy. Remove button if it is? -->
					<c:if test="${isUser}">
						<c:if
test="${isDefaultFolder.get(status.index)}">
							<div id="${folderID}"
class="remove_default">
								<h2><img src="<c:url
value="/resources/images/red-x4.png" />" title="Add to your Default Folders"
style="width:11px;"></h2>
							</div>
						</c:if>
						<c:if test="${not
isDefaultFolder.get(status.index)}">
							<div id="${folderID}"
class="add_default">
								<h2><img src="<c:url
value="/resources/images/plus.png" />" style="width:20px;"></h2>
							</div>
						</c:if>
					</c:if>
					<div id="${folderID}" class="apps_link">
						<h2><img src="<c:url
value="/resources/images/assistments.png" />" title="${folderNames.get(status.index)}"
style="width:90px;"></h2>
						<div class ="folder_name">
							<h3>${folderNames.get(status.index)}</h3>
						</div>
					</div>
				</div>
			</c:forEach>
			<c:forEach items = "${ProblemSetIds}" var = "ProblemSetId" varStatus =
"status">
				<div class="col-md-2 col-md-offset-1">
					<div id="${ProblemSetId}" class="apps_link2" title =
"${ProblemSetNames.get(status.index)}">
```

```
                                        <h2><img src="<c:url
value="/resources/images/view_problems_magnifier.png" />" style="width:90px;"></h2>
                                        <h3>${ProblemSetNames.get(status.index)}</h3>
                                    </div>
                                </div>
                            </c:forEach>
                    </div>
                    </div>
        </div>

        <div class="container " >

                <hr>

                <footer>
                <p>&copy; ASSISTments 2017</p>
                </footer>
        </div>
        <!-- /container -->
<div class="modal fade" id="message_modal">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
aria-hidden="true">&times;</span></button>
        <h4 class="modal-title">Message</h4>
      </div>
      <div class="modal-body">
        <p id="message_body"></p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
      </div>
    </div><!-- /.modal-content -->
  </div><!-- /.modal-dialog -->
</div><!-- /.modal -->
</body>
</html>
```

Index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Find and Assign</title>


<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/bootstrap.min.css" />" />
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/bootstrap-theme.min.css" />" />
<link rel="shortcut icon"
  href="<c:url value="/resources/images/shortcut_logo.png" />" />
```

```
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/jquery-ui.css" />" />

<script
	src="<c:url value="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"
/>"></script>
<script
	type = "text/javascript" src="<c:url value="/resources/js/bootstrap.min.js"
/>"></script>
<script type="text/javascript"
		src="<c:url value="/resources/js/navbar.js" />"></script>
<style type="text/css">
body {
	padding-top: 50px;
	padding-bottom: 20px;
}
.apps_link:hover{
	cursor:pointer;
}
.container_customize{
	padding:0 60px 0 30px;
}
a:hover{
	cursor:pointer;
}
.remove_default:hover{
	cursor:pointer;
}
.remove_default {
	position:absolute;
	left:145px;
	bottom:calc(100% - 35px);
}

.problem_building_img :hover {
	cursor:pointer;
}
.nav_active{
	color:white !important;
}
.footer_section{
	position:fixed;
	bottom:10px;
	text-align:center;
}

.apps_link {
	text-align: center;
	position: relative;
}
</style>
<script>
var CLASSROOM_CLIENT_ID =
"757036402283-8o3nu8pdve8snhj8ds11te8bnsrnmuu6.apps.googleusercontent.com";
var SCOPES = ["https://www.googleapis.com/auth/classroom.profile.emails",
		"https://www.googleapis.com/auth/classroom.rosters.readonly",
		"https://www.googleapis.com/auth/classroom.courses"];
var toolSelected = "";
	$(function(){
		$(".apps_link").click(function(){
```

```
                        var id = $(this).attr("id");
                        toolSelected = id;
                        locationStr = "${requestScope.location }";
                        locationStr = encodeURIComponent(locationStr);
                        if(id == "assistments_certified"){
                                window.location.assign("/FindAndAssign/path/1"+locationStr);
                        }else if (id == "engageNY"){
                                window.location.assign("/FindAndAssign/path/254673"
                                                +locationStr);
                        }else if (id == "chemistry_link"){

window.location.assign("/direct/SkillBuilderGoogleClassroom?folder_id=226695&tool_type=chemi
stry&location="
                                                +locationStr);
                        } else if(id == "problem_solving_link") {

window.location.assign("/direct/SkillBuilderGoogleClassroom?folder_id=231370&tool_type=probl
em_solving&location="+
                                                locationStr);
                        }else if(id == "login_button"){
                                window.location.assign("/FindAndAssign/login");
                        }else{
                                window.location.assign("/FindAndAssign/path/" + id);
                        }

                });


                $(".remove_default").click(function(){
                                var id = $(this).attr("id");
                                window.location.assign("/FindAndAssign/removeDefault/" + id);
                });
                $(".apps_link").hover(function(){
                        $("img",this).animate({
                                width:"+=6"
                        },100);
                },function(){
                        $("img",this).animate({
                                width:"-=6"
                        },100);
                });
                $(".remove_default").hover(function(){
                        $("img",this).animate({
                                width:"+=2"
                        },100);
                },function(){
                        $("img",this).animate({
                                width:"-=2"
                        },100);
                });
                $(".problem_building_img").hover(function() {
                        $("img",this).animate({
                                width:"+=8",
                                height: "+=8"
                        },100);
                },function(){
                        $("img",this).animate({
                                width:"-=8",
                                height: "-=8"
                        },100);
```

```
		});
	});

	function imageClick() {
		gapi.auth.authorize({
			'client_id': CLASSROOM_CLIENT_ID,
			'scope': SCOPES,
			'immediate': true}, handleAuthResult);
	}

	function handleAuthResult(authResult) {
		if (authResult && !authResult.error) {
			var firstName;
			var lastName;
			var emailAddr;
			var ownerId;
			//get user profile
		    var request = gapi.client.request({
				root : 'https://classroom.googleapis.com',
				path : 'v1/userProfiles/me',
			});

			request.execute(function(resp) {
				if(resp.error != null) {
				if(resp.error.code == 403) {
					$("#message_body").html("Sorry... Google Classroom is
not available for your google account at this time.");
					$("#message_body").css("color", "blue");
					$('#message_modal').modal('show');
					return;
				}
				$("#message_body").html("Sorry... Something goes wrong here
with Google Classroom. But we have no idea what causes the problem.");
				$("#message_body").css("color", "red");
				$('#message_modal').modal('show');
				return;
			}

				firstName = resp.name.givenName;
				lastName = resp.name.familyName;
				emailAddr = resp.emailAddress;
				ownerId = resp.id;
				$.ajax({
					url: 'google_classroom/go_to_assistments',
					type: 'POST',
					data: {google_user_id: ownerId, first_name: firstName,
last_name: lastName, email_address: emailAddr},
					async: false,
					success: function(data) {
						window.location.replace(data.location);
					},
					error: function(data) {
						$("#message_body").html("Sorry... Something goes
wrong here. We cannot direct you to assistments.org."+
                                                "Please email us at
assistments-help@wpi.edu with this error.");
					$("#message_body").css("color", "blue");
					$('#message_modal').modal('show');
					}
				});
```

```
                        });
                } else {
                        $("#message_body").html("It seems that you don't log into Google.
Please sign into Google first and then access the app.");
                $("#message_body").css("color", "blue");
                $('#message_modal').modal('show');
                }
        }
</script>
</head>
<body>
        <jsp:include page="navbar.jsp"/>

        <!-- Main jumbotron for a primary marketing message or call to action -->
        <div class="jumbotron" style="background-color:white;">
                <div class="container">
                        <div style="text-align: center;">
                        <h1><small style="color:#333;">Find and Assign Explorer</small></h1>
                        <h2 style="color:#777;">Your Starting Folders</h2>
                        </div>
                        <div class="row" style="margin-top:70px;">
                        <div class="col-md-2 col-md-offset-1">
                        <div id="assistments_certified" class="apps_link">
                                <h2><img src="<c:url value="/resources/images/assistments.png"
/>" style = "width:90px"/></h2>
                                <h3>ASSISTments Certified</h3>
                        </div>
                        </div>
                        <c:forEach items = "${folderIds}" var = "folderID" varStatus =
"status">
                                <div class="col-md-2 col-md-offset-1">
                                        <div id="${folderID}" class="remove_default">
                                                <h2><img src="<c:url
value="/resources/images/red-x4.png" />" style="width:10px;"></h2>
                                        </div>
                                        <div id="${folderID}" class="apps_link">
                                                <h2><img src="<c:url
value="/resources/images/assistments.png" />" style="width:90px;"></h2>
                                                <h3>${folderNames.get(status.index)}</h3>
                                        </div>

                                </div>
                        </c:forEach>
                </div>
                </div>
        </div>

        <div class="container " >

                <hr>

                <footer>
                <p>&copy; ASSISTments 2017</p>
                </footer>
        </div>
        <!-- /container -->
<div class="modal fade" id="message_modal">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
```

```
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
aria-hidden="true">&times;</span></button>
        <h4 class="modal-title">Message</h4>
      </div>
      <div class="modal-body">
        <p id="message_body"></p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
      </div>
    </div><!-- /.modal-content -->
  </div><!-- /.modal-dialog -->
</div><!-- /.modal -->
</body>
</html>
```

navbar.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1">


<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/bootstrap.min.css" />" />
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/bootstrap-theme.min.css" />" />
<link rel="shortcut icon"
  href="<c:url value="/resources/images/shortcut_logo.png" />" />
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/jquery-ui.css" />" />
  <script
        src="<c:url value="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"
/>"></script>
<style type="text/css">
.navbar_item:hover{
        cursor:pointer;
}
</style>
<script>
$(function(){
        $(".navbar_item").click(function(){
                var id = $(this).attr("id");
                locationStr = encodeURIComponent("${requestScope.location }");
                if(id == "landing_page" || id == "logo"){
                        toolSelected = "landing_page";
                        window.location.assign("/FindAndAssign/home");
                }else if(id == "login_button"){
                        toolSelected = "login_link";
                        var link = "/FindAndAssign/login";
                        link = link + "?callbackURL=" + window.location.href;
                        window.location.assign(link);
                }else if(id == "logout_button"){
```

```
                        toolSelected = "logout_link";
                        window.location.assign("/FindAndAssign/logout");
                }else if(id == "hello_user_link"){
                        toolSelected = "user_link";
                        window.location.assign("/FindAndAssign/userInfo");
                }else if(id == "custom_problems_link"){
                        toolSelected = "problems_link"
                        window.location.assign("/FindAndAssign/ViewCustomProblems")
                }
        });
});
</script>
</head>
<body>
<nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container_customize">
                <div class="navbar-header">
                        <button type="button" class="navbar-toggle collapsed"
                                data-toggle="collapse" data-target="#navbar"
aria-expanded="false"
                                aria-controls="navbar">
                                <span class="sr-only">Toggle navigation</span>
                                <span class="icon-bar"></span>
                                <span class="icon-bar"></span>
                                <span class="icon-bar"></span>
                        </button>
                        <a class="navbar-brand navbar_item" id="logo"><img src="<c:url
value="/resources/images/assistments.png"/>" style="width:30px;
display:inline;margin-right:5px;"/>Find And Assign</a>
                </div>
                <div id="navbar" class="navbar-collapse collapse">
                        <c:if test="${!isUser}">
                                <ul class="nav navbar-nav">
                                        <li><a id="login_button" class="navbar_item">Log In</a>
                                </ul>
                        </c:if>
                        <c:if test="${isUser}">
                                <ul class="nav navbar-nav">
                                        <li><a id="hello_user_link" class="navbar_item">Hello,
${userDisplayName}</a>
                                        <li><a id="logout_button" class="navbar_item">Log
Out</a>
                                        <c:if test="${isTeacher}">
                                                <li><a id="assignments_button"
class="navbar_item">View Assignments</a>
                                                <li><a id="custom_problems_link"
class="navbar_item">Custom Problem Sets</a>
                                        </c:if>
                                </ul>
                        </c:if>
                </div>
                <!--/.navbar-collapse -->
        </div>
        </nav>
</body>
</html>
```

problems.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
	pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Find And Assign Problem Preview</title>
</head>
<style type="text/css">
hr {
	margin: 5px;
	border-top: 1px solid black !important;
}
.add_button{
	background-color: #4CAF50;
	border:none;
	color:white;
	text-align: center;
	font-size: 16px;
}
.remove_button{
	background-color: #F44336;
	border:none;
	color:white;
	text-align: center;
	font-size: 16px;
}

</style>
<body>

	<script
		src="<c:url
value="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js" />"></script>
	<script type="text/javascript"
		src="<c:url value="/resources/js/bootstrap.min.js" />"></script>
	<script
		src="<c:url value="https://apis.google.com/resources/js/platform.js"
/>"></script>
	<script
		src="<c:url value="https://apis.google.com/resources/js/client.js"
/>"></script>
	<script>
	$(function(){
		$(".add_button").click(function(){
			var id = $(this).attr("id");
			window.location.assign("/FindAndAssign/addToCart/" + id);

		});
		$(".remove_button").click(function(){
			var id = $(this).attr("id");
			window.location.assign("/FindAndAssign/removeFromCart/" + id);

		});
	});
	//$(window).scroll(function () {
	    //set scroll position in session storage
	  //  sessionStorage.scrollPos = $(window).scrollTop();
```

```
        //});
        //var init = function () {
            //get scroll position in session storage
          //  $(window).scrollTop(sessionStorage.scrollPos || 0)
        //};
        //window.onload = init;
        </script>


        <c:forEach items="${problemMap}" var="problem" varStatus="status">
        <hr style="margin: 30px 0 0 0;">
        <c:if test = "${isTeacher}">
                 <c:if test = "${not problem.value.inCart}">
                        <button class="add_button" id="${problem.value.id}">Add to Custom
Problem Set</button><br>
                </c:if>
                <c:if test = "${problem.value.inCart}">
                        <button class="remove_button" id="${problem.value.id}">Remove from
Custom Problem Set</button><br>
                </c:if>
                <c:if test="${problem.value.inCart}">

                </c:if>
        </c:if>
        <!-- ${status.index + 1}.-->
        ${problem.value.problemType}<br>
        ${problem.value.problemText}<br>
                <c:choose>
                        <c:when test="${problem.value.problemType == 'Multiple Choice'}">
                                <!-- choose 1 -->
                                <c:forEach items="${problem.value.answers}" var="answerMap"
                                        varStatus="answerStatus">
                                        <p>
                                                <input type="radio"
name="${problem.value.problemText}"> ${answerMap.answerText}
                                        </p>
                                </c:forEach>
                        </c:when>
                        <c:when test="${problem.value.problemType == 'Check All That Apply'}">
                                <!-- choose N -->
                                <c:forEach items="${problem.value.answers}" var="answerMap"
                                                varStatus="answerStatus">
                                        <p>
                                                <input type="checkbox"> ${answerMap.answerText }
                                        </p>
                                </c:forEach>
                        </c:when>
                        <c:when test="${problem.value.problemType == 'Rank'}">
                                <!-- rank -->
                                <c:forEach items="${answerMapList}" var="answerMap"
                                        varStatus="answerStatus">
                                        <p>
                                                <input type="text" style="width: 2em;">
${answerMap.answerText }
                                        </p>
                                </c:forEach>
                        </c:when>
                        <c:when test="${problem.value.problemType == 'Open Response'}">
                                <!-- open response -->
                                <textarea></textarea>
```

```
                        </c:when>
                        <c:otherwise>
                                <input type="text">
                        </c:otherwise>

                </c:choose>
                <!--<c:forEach items="${answerMapList}" var="answerMap"
                        varStatus="answerStatus">
                ${answerMap.answerText}<br>

                </c:forEach>-->

        </c:forEach>
<hr style="margin: 30px 0 0 0;">
</body>
</html>
```

problemSets.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<script
        src="<c:url value="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"
/>"></script>
<script type="text/javascript"
        src="<c:url value="/resources/js/bootstrap.min.js" />"></script>
<script
        src="<c:url value="https://apis.google.com/resources/js/platform.js" />"></script>
<script
        src="<c:url value="https://apis.google.com/resources/js/client.js" />"></script>
<title>Problem Sets in ${parentName}</title>
<link href="<c:url value="/resources/stylesheets/bootstrap.min.css"/>"
        rel="stylesheet">
<link
        href="<c:url value="/resources/stylesheets/bootstrap-tour.min.css"/>"
        rel="stylesheet">
<link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-awesome.min.css">
<script type="text/javascript"

src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-select/1.7.5/js/bootstrap-select.min.j
s"></script>
</head>
<body>
        <jsp:include page="navbar.jsp"/>
        <table class='table table-hover' style="margin-top: 50px;">
                <thead style='font-size: 0.8em;'>
                        <tr class='row'>
                                <th class='col-md-1' style='text-align: right;'>View</th>
                                <th class='col-md-1' style='text-align: left;'>Assign</th>
                                <th class='col-md-10'></th>
                        </tr>
```

```
            </thead>
            <tbody style='font-size: 1.5em;'>
                    <c:forEach items="${ProblemSetIds}" var="problemSetID"
                            varStatus="status">
                            <tr class='curriculum_item row'>
                                    <td class='col-md-1' style='text-align: right;'>
                                            <div class="btn-group onclick" style="cursor:
pointer;">
                                                    <a class='dropdown-toggle'
data-toggle="dropdown"
                                                            aria-haspopup="true"
aria-expanded="true"> <img
                                                            src="<c:url
value='/resources/images/view_problems_magnifier.png'/>"
                                                            style='width: 30px'></a>
                                                    <ul class='dropdown-menu pull-middle'>
                                                    <c:if test="${isTeacher}">
                                                            <li> <a class='view_problem_icon'
target='_blank'
                                                                    href="<c:url
value='/problems/${problemSetID}'/>">View All Problems</a></li>
                                                    </c:if>
                                                            <li><a class='student_view'
target='_blank'
                                                                    href='<c:url
value='/problems/${problemSetID}?Special=true'/>'>View Problem Sample</a></li>
                                                    </ul>
                                            </div>
                                    </td>
                                    <td class='col-md-1' style='text-align: left;'>
                                            <div class="btn-group onclick" style="cursor:
pointer;">
                                                    <a class='dropdown-toggle'
data-toggle="dropdown"
                                                            aria-haspopup="true"
aria-expanded="true">
                                                            <img
src='//www.gstatic.com/classroom/logo_square_48.svg' style='width: 25px; height: 25px'></a>
                                                    <ul class='dropdown-menu pull-middle'>
                                                            <li><a class='view_problem_icon'
target='_blank'
                                                                    href="<c:url
value='/assign/${problemSetID}'/>">Assign to ASSISTments</a></li>
                                                            <li><a class='student_view'
target='_blank'
                                                                    href="<c:url
value='/assignToLms/${problemSetID}?lmsPtype=1'/>">Assign to Google Classroom</a></li>
                                                    </ul>
                                            </div>
                                    </td>
                                    <td
class='col-md-10'>${ProblemSetNames.get(status.index) }</td>
                            </tr>
                    </c:forEach>
            </tbody>
        </table>
</body>
</html>
```

view_custom_problems.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
	pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<!DOCTYPE html>
<!--  PUBLIC "-//W3C//DTD HTML 5.01 Transitional//EN"
"http://www.w3.org/TR/html5/loose.dtd"> -->
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Your Custom Problem Sets</title>
</head>
<style type="text/css">
.remove_button {
	background-color: #F44336;
	border: none;
	color: white;
	text-align: center;
	font-size: 16px;
}

.publish_button {
	float: right;
	font-size: 20px;
	border-radius: 4px;
}

.just_publish {
	background-color: #89cff0;
	color: black;
	border: 2px solid #0000FF;
	margin-left: auto;
	margin-right: 5%;
	margin-top: 10px;
}

.publish_and_assign {
	background-color: #00FF00;
	color: black;
	border: 2px solid #228B22;
	margin-left: auto;
	margin-right: 5%;
	margin-top: 10px;
}
</style>
<script
	src="<c:url value="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"
/>"></script>
<script type="text/javascript"
	src="<c:url value="/resources/js/bootstrap.min.js" />"></script>
<script
	src="<c:url value="https://apis.google.com/resources/js/platform.js" />"></script>
<script
	src="<c:url value="https://apis.google.com/resources/js/client.js" />"></script>
<script>
	$(function() {
```

```
		$(".remove_button").click(function() {
			var id = $(this).attr("id");
			window.location.assign("/FindAndAssign/removeFromCart/" + id);

		});
		$(".just_publish").click(function() {
			var name = $("#cart_name").val();
			name = encodeURIComponent(name);
			//var description = $("#cart_description").val();
			var url = "/FindAndAssign/publishSet?Name=" + name;
			/*if (description != null) {
				url += "&Description=" + description;
			}*/
			window.location.assign(url);
		})
		$('.publish_button').attr('disabled', true);
		$('#cart_name').keyup(function() {
			if ($(this).val.length != 0) {
				$('.publish_button').attr('disabled', false);
			} else {
				$('.publish_button').attr('disabled', true);
			}
		});

	});
	//$(window).scroll(function () {
	//set scroll position in session storage
	//  sessionStorage.scrollPos = $(window).scrollTop();
	//});
	//var init = function () {
	//get scroll position in session storage
	//  $(window).scrollTop(sessionStorage.scrollPos || 0)
	//};
	//window.onload = init;
</script>

<body>
	<jsp:include page="navbar.jsp" />
	<div id="page" style="padding: 50px 0px 0px 0px;">

		<div id="shopping_cart" style="padding: 0px 0px 0px 10%;">

			<c:if test="${fn:length(problemMap) != 0}">
				<h3>Shopping Cart:</h3>
				<br>
	Click the add button next to any problem within any problem set to add it to your
shopping cart.<br>
				<div class="problems" style="width: 90%; display: inline-block;
border: medium solid;">
					Name: <input id="cart_name" type="text"
						placeholder="Name your custom problem set"
style="width: 300px;">
					<!-- Description:
					<textarea id="cart_description" rows="6" cols="50"
						style="vertical-align: top;"></textarea>  No
need for Problem Set Descriptions.-->
					<button class="publish_button
just_publish">Publish</button>
					<button class="publish_button
publish_and_assign">Publish
```

```
                                        and Assign</button>
                                    <c:forEach items="${problemMap}" var="problem"
varStatus="status">
                                        <hr style="margin: 30px 0 0 0;">
                                        <button class="remove_button"
id="${problem.value.id}">Remove
                                                from Custom Problem Set</button>
                                        <br>

                                        <!-- ${status.index + 1}.-->
        ${problem.value.problemType}<br>
        ${problem.value.problemText}<br>
                                        <c:choose>
                                            <c:when test="${problem.value.problemType
== 'Multiple Choice'}">
                                                <!-- choose 1 -->
                                                <c:forEach
items="${problem.value.answers}" var="answerMap"
                                                    varStatus="answerStatus">
                                                    <p>
                                                        <input
type="radio"> ${answerMap.answerText}
                                                    </p>
                                                </c:forEach>
                                            </c:when>
                                            <c:when
                                                test="${problem.value.problemType
== 'Check All That Apply'}">
                                                <!-- choose N -->
                                                <c:forEach
items="${problem.value.answers}" var="answerMap"
                                                    varStatus="answerStatus">
                                                    <p>
                                                        <input
type="checkbox"> ${answerMap.answerText }
                                                    </p>
                                                </c:forEach>
                                            </c:when>
                                            <c:when test="${problem.value.problemType
== 'Rank'}">
                                                <!-- rank -->
                                                <c:forEach
items="${answerMapList}" var="answerMap"
                                                    varStatus="answerStatus">
                                                    <p>
                                                        <input type="text"
style="width: 2em;">

${answerMap.answerText }
                                                    </p>
                                                </c:forEach>
                                            </c:when>
                                            <c:when test="${problem.value.problemType
== 'Open Response'}">
                                                <!-- open response -->
                                                <textarea></textarea>
                                            </c:when>
                                            <c:otherwise>
                                                <input type="text">
                                            </c:otherwise>
```

```
                                        </c:choose>
                                        <!--<c:forEach items="${answerMapList}"
var="answerMap"
                        varStatus="answerStatus">
                ${answerMap.answerText}<br>

                </c:forEach>-->

                                </c:forEach>

                        </div>
                </div>
                </c:if>
                <h3>
                        Your<c:if test="${fn:length(problemMap)!= 0}"> Other </c:if> Custom
Problem Sets:
                </h3>
                <div>
                        <table class='table table-hover' style="margin-top: 50px;">
                                <thead style='font-size: 0.8em;'>
                                        <tr class='row'>
                                                <th class='col-md-1' style='text-align:
right;'>View</th>
                                                <th class='col-md-1' style='text-align:
left;'>Assign</th>
                                                <th class='col-md-10'></th>
                                        </tr>
                                </thead>
                                <tbody style='font-size: 1.5em;'>
                                        <c:forEach items="${CustomSetIds}" var="customSetID"
                                                varStatus="status">
                                                <tr class='curriculum_item row'

title="${CustomSetDescriptions.get(status.index) }">
                                                        <td class='col-md-1' style='text-align:
right;'>
                                                                <div class="btn-group onclick"

style="cursor: pointer;">
                                                                        <a class='view_custom_set'
                                                                                href="<c:url
value='/customProblem/${customSetID}'/>"> <img
                                                                                src="<c:url
value='/resources/images/view_problems_magnifier.png'/>"
                                                                                style='width:
30px'></a>
                                                                </div>
                                                        </td>
                                                        <td class='col-md-1' style='text-align:
left;'>
                                                                <div class="btn-group onclick"

style="cursor: pointer;">
                                                                        <a class='dropdown-toggle'

data-toggle="dropdown"

aria-haspopup="true" aria-expanded="true"> <img

src='//www.gstatic.com/classroom/logo_square_48.svg'
                                                                                style='width: 25px;
height: 25px'></a>
```

```
                                                                    <ul class='dropdown-menu
pull-middle'>
                                                                        <li><a
class='view_problem_icon' target='_blank'
                                                                            href="<c:url
value='/assign/${customSetID}'/>">Assign to
ASSISTments</a></li>
                                                                        <li><a
class='student_view' target='_blank'
                                                                            href="<c:url
value='/assignToLms/${customSetID}?lmsPtype=1'/>">Assign
                                                                                to
Google Classroom</a></li>
                                                                    </ul>
                                                            </div>
                                                    </td>
<td class='col-md-10'>${CustomSetNames.get(status.index) }</td>
</tr>
</c:forEach>
</tbody>
</table>
</div>
</div>
</body>
</html>
```

view_problems.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%@ taglib prefix="c"  uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title><c:if test="${not custom}"> View Problems ></c:if><c:if test="${custom}"> Custom
Problem Set ></c:if> "${problemSetName}"</title>
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/bootstrap.min.css" />" />
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/bootstrap-theme.min.css" />" />
<link rel="shortcut icon"
  href="<c:url value="/resources/images/shortcut_logo.png" />" />
<link rel="stylesheet"
  href="<c:url value="/resources/stylesheets/jquery-ui.css" />" />
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
<!-- Optional theme -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap-theme.min.css">
<!-- Latest compiled and minified JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
</head>
<body>
              <jsp:include page="navbar.jsp"/>
              <div class="container" style="text-align: left; color: black; padding: 50px
```

```
30px 30px 30px; background: white;">
                        <c:if test="${custom}">
                                <h4>Custom Problem Set: ${problemSetName}</h4>
                        </c:if>
                        <c:if test="${sessionScope.is_skill_builder }">
                                <h4>Five selected problems from
"${requestScope.problem_set_name }"</h4>
                        </c:if>
                        <c:if test="${not sessionScope.is_skill_builder && not custom}">
                                <c:if test="${not isTeacher}">
                                        <h5>This is the Student View. Log in with a teacher
account to see all problems.</h5>
                                </c:if>
                                <h4>Problem Set ${problemSet.getName()}:
"${problemSetName}"</h4>
                        </c:if>
                        <jsp:include page="problems.jsp" />

                        <div style="height: 65px; margin-top: 15px;">

                        <footer>
                <p>&copy; ASSISTments 2017</p>
                </footer>
                        </div>
                </div>

</body>
</html>
```

breadcrumb.css

```
/* Style the list */
ul.breadcrumb {
    padding: 10px 16px;
    list-style: none;
    background-color: #eee;
    font-size: 17px;
}

/* Display list items side by side */
ul.breadcrumb li {
    display: inline;
}

/* Add a slash symbol (/) before/behind each list item */
ul.breadcrumb li+li:before {
    padding: 8px;
    color: black;
    content: ">\00a0";
}

/* Add a color to all links inside the list */
ul.breadcrumb li a {
    color: #0275d8;
    text-decoration: none;
}

/* Add a color on mouse-over */
```

```
ul.breadcrumb li a:hover {
    color: #01447e;
    text-decoration: underline;
}
```

navbar.js

```
$(".navbar_item").click(function(){
               var id = $(this).attr("id");
               locationStr = encodeURIComponent("${requestScope.location }");
               if(id == "landing_page" || id == "logo"){
                      toolSelected = "landing_page";
                      window.location.assign("/FindAndAssign/home");
               }else if(id == "login_button"){
                      toolSelected = "login_link";
                      var link = "/FindAndAssign/login";
                      link = link + "?callbackURL=" + window.location.href;
                      window.location.assign(link);
               }else if(id == "logout_button"){
                      toolSelected = "logout_link";
                      window.location.assign("/FindAndAssign/logout");
               }else if(id == "hello_user_link"){
                      toolSelected = "user_link";
                      window.location.assign("/FindAndAssign/userInfo");
               }
       });
```