

Spring 2020 Senior Project

Title of the Project: ChordScore

Team members (3-5):

- | | | |
|---|--------------------------------|----------------------|
| · | Name: Ruth Bearden (team lead) | Student ID: rbeard13 |
| · | Name: Zsavaughn Daniel | Student ID: zdaniel5 |
| · | Name: Daisy Hernandez | Student ID: dherna55 |
| · | Name: Austin Roberts | Student ID: arobe176 |
| · | Name: Tadiwa Mangadze | Student ID: tmangadz |

Project Outline (Report #0 Group #1) – 1/12/2020 – 1/26/2020

Summary - ChordScore Grader application for Music Theory I student homework

The goal of this project is to build a web application to automatically scan and assign a grade to Music Theory I student homework. The idea for this project came from Dr. Jeffrey Yunek, Music Theory professor at KSU. His hope is to have a working and tested application by the end of the Spring 2020 semester that he can use in a highly supervised way for the Fall 2020 semester Music Theory I class.

Whether or not the ChordScore Grader application is able to accurately assign grades to student homework, the study can provide useful information to the field of Music Theory Pedagogy research in which the idea of automatic homework grading seems yet to be explored.

Requirements

Homework submission and grading process

The ChordScore Grader application will be a web application that facilitates the following steps

1. Students create separate application accounts
2. Students submit their homework as scanned image files through these accounts
1. ChordScore compares answers extracted from student homework to answers from the professor's "answer key" homework sheet
2. ChordScore assigns a grade to the homework and marks "incorrect" answers with a red box
3. Professor logs into a grader account and views the homework that ChordScore has graded and marked
4. Professor either accepts or modifies grade ChordScore generated

Application UI - Student

ChordScore's *student account* web interface must allow students to create an account, submit scanned homework images, and view auto-generated grades and mark-ups (note: this requirement may be subject to change).

Application UI - Professor

ChordScore's *grader account* web interface must allow the professor to submit a homework answer key, define a set of grading parameters (i.e. points per answer in a given section of the homework document), and allow the professor to review auto-graded student homework submissions.

Grade generation

Grade generation will be based on a *correct answer: total answer* ratio defined by the professor and scaled to the number of points the professor assigns a given section of the homework (region of the answer key image). Each element of student homework that does not match the corresponding answer key answer (whether a short answer or music notation answer) will be boxed in red and points will be deducted from the student's grade. This will enable the professor upon reviewing the marked-up homework submission to determine whether or not the auto-generated grade is correct and make appropriate adjustments.

Grade accuracy evaluation

For at least a few weeks of the Spring 2020 semester, this application will be monitored by ChordScore's development team to assess its accuracy, including its false negative (FN) rate and false positive (FP) rates, where a "negative" is a correct answer and "positive" and incorrect answer. Our goal is to reach at least a 95% accuracy rate which is considered a standard level of accuracy significance in scientific research. However, it is also important to attain a low FP rate as it is more desirable, per request by Dr. Yunek, to miss a few errors and include an "automatic curve" than to identify correct answers as incorrect.

Accuracy will be measured using *precision*, *recall*, and *f-measure* standard machine learning metrics for multiclass classification.

According to Machine Learning Mastery^[3], these metrics are defined as follows (where c = class):

$$\text{Precision} = \text{Sum } c \text{ in } C \text{ TruePositives}_c / \text{Sum } c \text{ in } C (\text{TruePositives}_c + \text{FalsePositives}_c)$$

$$\text{Recall} = \text{Recall} = \text{Sum } c \text{ in } C \text{ TruePositives}_c / \text{Sum } c \text{ in } C (\text{TruePositives}_c + \text{FalseNegatives}_c)$$

$$\text{F-measure} = (2 * \text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

If ChordScore can attain sufficiently high accuracy scores in the Spring 2020 semester, Dr. Yunek will use this application (in a highly supervised way) for his Fall 2020 Music Theory I class. The end goal is to increase the application's grade accuracy to the point where it can run on its own with minimal supervision such that Dr. Yunek need only manually review grades that deviate significantly from the expected grades that could be due to incorrect answer detection on the part of ChordScore.

Methods

Compose a Phrase Using the G Major Scale
www.musictechteacher.com

Name: _____ Date: _____

A 'phrase' in music is similar to forming a question and answer. Usually the answer ends on the tonic or first note in the scale. The question usually ends on another pitch, such as the dominant or fifth note in the scale. The melody and rhythms may be similar in the question and answer of the phrase.

Using half notes, write the notes to the G Major Scale on the staff below. Rehearse the notes on the keyboard to the G Major Scale.

Using the notes from the G Major Scale, create a four measure musical phrase in the form of a question. When you have completed your question, trade papers with another student and have them complete your question with a musical answer. (If you have Sibelius software, enter the notes using the keyboard.)

Question

Answer

Does each person in the group:

- 1) understand what a musical phrase is? (5 points)
- 2) write the notes to the G Major Scale correctly on the staff? (15 points)
- 3) complete the question and answer portion of the phrase? (30 points)

Optional:

- 4) notate the music using Sibelius software? (25 points)
- 5) play the musical phrase using the keyboard or singing the phrase correctly? (25 points)

Students may receive 50 or 100 points total, depending on the availability of notation software and the music being performed by the student.

1) _____
2) _____
3) _____
4) _____
5) _____
TOTAL = _____

© www.musictechteacher.com

Figure 1 – Fields on homework sheet as defined by instructor for an assignment

Image pre-processing

1. Submitted homework images will require preprocessing before answers can be read and graded. Either through manual input from the professor or through automatic image segmentation (to be determined through experimentation), fields on each image will be identified and categorized as the following: Irrelevant text (Fig. 1 - Yellow)
2. Free response (Fig. 1 - Blue)
3. Music notation response (Fig. 1 - Red)
4. Short answer response (Fig. 1 - Green)

Image processing by field

1. *Irrelevant text* - ignored
2. *Free response* - answers will be extracted as a series of characters using handwriting recognition algorithms
3. *Music notation response* - answers will be converted to a digital music notation format (probably MusicXML) using CNN and possibly RNN algorithms along with other methods used in deep learning-based OMR (Optical Music Recognition) research.^[2]

4. *Short response* - same as *Free response*

Background and Related Work

Automatic graders for handwritten homework using predefined answer fields is not a novel idea and has been used in the Bakpax (Beta) ^[1] mobile product which suggests that others believe it to be a viable application of handwriting recognition algorithms. Creating an auto grader for music staff notation will add a dimension of handwritten answer recognition that falls under the field of Optical Music Recognition (OMR). The algorithms that will be used to create a music recognition model will be based on the current methods considered in recent research papers. The CNN (Convolutional Neural Network), RNN (Recurrent Neural Network), and LSTM (Long Short-Term Memory) algorithms currently dominate the OMR research field according to Huang et al. in the 2019 paper, *State-of-the-Art Model for Music Object Recognition with Deep Learning* ^[2]. Methods used will most likely be similar to those used in this paper including methods for data collection. This paper uses Musescore as a data source since this website provides music files with both images from music PDF formats and MusicXML (a digital music notation file format). The study performed by Huang et al. used this image to digital format correspondence to train the neural network algorithms used in the study.

References

- ^[1] Bakpax. <https://www.bakpax.com/#how-it-works>
- ^[2] Z. Huang, X. Jia, Y. Guo. "State-of-the-Art Model for Music Object Recognition with Deep Learning", *Applied Sciences* (2019). Retrieved from <https://www.mdpi.com/2076-3417/9/13/2645>, 12 Jan. 2019.
- ^[3] <https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/>

Report #1 Group #1 – 1/26/2020 – 2/9/2020

1. Conventions/ Definitions

We worked on the conventions and definitions that we would be using for this application. These are the conventions:

For all applications, bold-type font will be used

- E.g. Microsoft Word will be displayed Microsoft Word

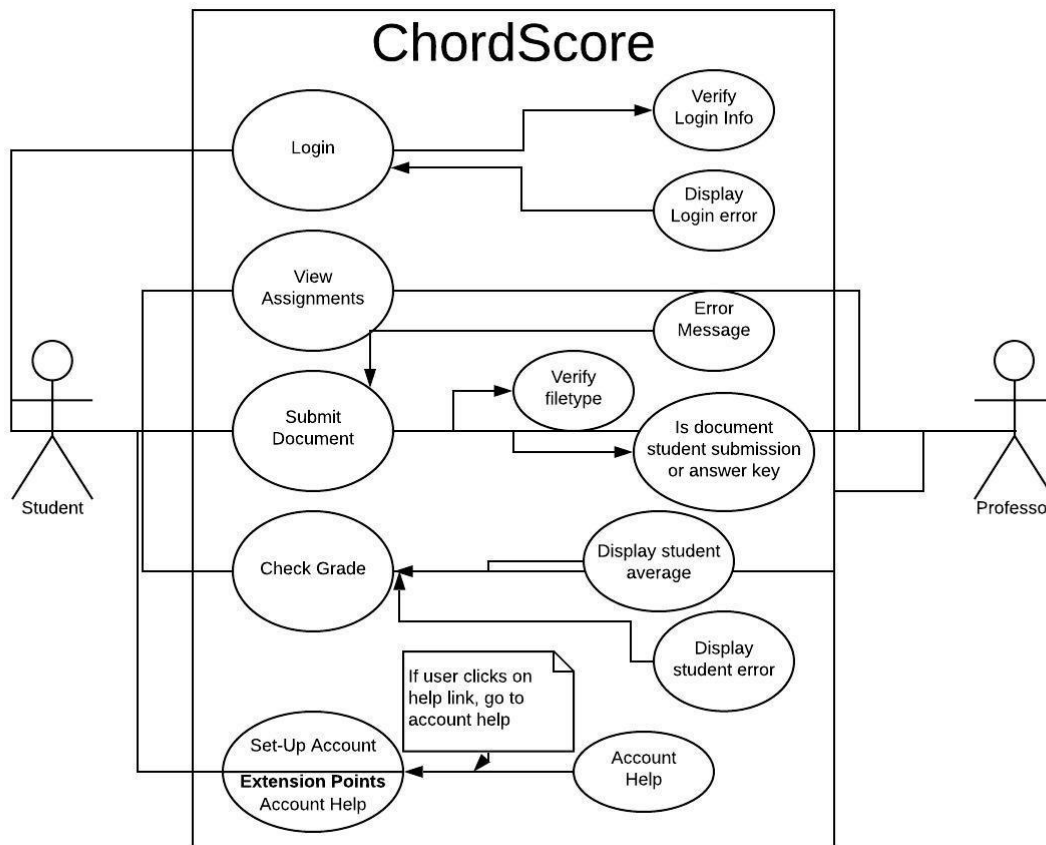
These are the definitions:

- In-House Testing- This testing will test the functional and non-functional requirements of the software.
- Manual Testing- This is internal testing that will be done on the program itself to improve and fix potential bugs in the code.
- User Testing- This is external testing that will be done using case studies to make the team aware of any potential flaws in the application by allowing users to interact with ChordScore.
- Regression Testing- This would test the program after any changes have been made in order to see if it affected the program.
- Integration Testing- This testing will make sure that there are no defects once two or more modules are combined.

We sought to identify the target audience and and to identify those who would benefit from this application. Our targets groups are students and instructors. Target users will be music students who will use ChordScore to upload their homework and analyze their grades. It will benefit the music professors by allowing them to use the application to virtually grade music student's homework via a scanned answer key. Another audience we discussed about during our group meetings, is that professional developers could analyze the project we've created to input difference ideas or even advance on it beyond our original purpose. We drew up mock samples and described how each document would serve its own purpose. There is the student image, a markup image, and an answer key. The answer key would derive from what the professor created, and this would be used against the student image. The markup image would highlight the wrong answers and the professor could then add onto some feedback to give the student. The student image is where the document that the student as made their answers on and the sheet is provided by the professor. From there, we discussed how we were going to create our data sample.

We worked on formulating each aspect of the application. This would provide a visual representation of what the expected result would be.

Use-Case Diagram:




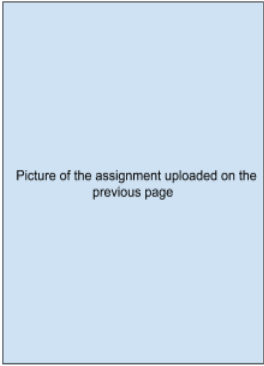
We worked on creating a use case diagram to allow us to break down each feature of the program and to break down how it would work. This was based on a means to start on pseudo-code by using the diagram which outlines the direct interactions between the student and the professor through the application. This was created before our meeting with Dr. Yunek, so we expect our diagram and plan to change some bit as he gives us more detail on what the application needs to do. An online diagram software was used to build this and it was hard to navigate at first but once we played around with it. It all came together nicely. From this, we also worked on setting deadlines for ourselves and our overall team.

2. Project Schedule


A Gantt chart was created that would help us navigate through the weeks as they come and that would allow us to stay on track the rest of the semester. We discuss what is needed for the group and what is needed for the application development. We divided the groups into three different sections to work on during the week which are the following:

3. Web Application Development

This prototype serves to design the functionality of the application. It provided the visual representation of what this would end up doing. There are different displays of each feature that the application will serve the user with. This is where the developers will start from and gather ideas on what the source code will look like.

	<div>ChordScore Logo</div> <div>Welcome Back! Please sign in:</div> <div>Email: <input type="text"/></div> <div>Password: <input type="password"/></div> <div><input type="button" value="Sign in"/></div> <div>Don't have an account? Create one here</div>		<div>ChordScore Logo</div> <div>Welcome to ChordScore!</div> <div>First Name: <input type="text"/></div> <div>Last Name: <input type="text"/></div> <div>Email: <input type="text"/></div> <div>Password: <input type="password"/></div> <div><input type="button" value="Sign up"/></div>	
<div>Both dates will be selected on a calendar, with a drop down menu to the right to select the time</div> <div>Submit takes the professor to a new page and updates the database with the given information</div>	<div>ChordScore Logo</div> <div><div>Name</div><div>Drop down menu containing: Class list Log out</div></div> <div>New Assignment</div> <div>Assignment Title: <input type="text"/></div> <div>Open Date: <input type="text"/></div> <div>Due Date: <input type="text"/></div> <div>Upload Answer Key: <input type="button" value="Browse"/></div> <div><input type="button" value="Submit"/></div>		<div>ChordScore Logo</div> <div>Assignment Title</div> <div><div>Assignment Title will display the title given on the previous page</div><div>Using the drop down menu, the Professor will select the type of field, then using the mouse, they will draw a box around the field they are trying to create.</div></div> <div></div> <div><div>Drop down menu for selection between new free response fields, new notation fields, and new short answer fields</div><div><input type="button" value="New Field"/></div></div>	

This page is the first page that gets pulled when a professor logs in


ChordScore Logo

Name

Drop down menu containing:
Class list
Log out

Class list:

Class name	Semester

Add New Class

ChordScore Logo

Name

Drop down menu containing:
Assignments
Log out

Assignment Title will display the title given when the assignment is created.

Assignment Title

Description of assignment with any download links to necessary assignment folders

Browse
Submit

4. Data Collection

We discussed and set a plan to gather handwritten music notation samples from students. In order to gather samples that are most similar to music notation found on student homework, we determined that music notation consisting of a mixture of handwritten and printed musical elements to be best data format. Different types of software were observed to pinpoint which one would allow the team to find music sheets. Music software, *Musescore*, was selected as this gave the option to save files in different file types. One file type that was needed is the XML filetype which *Musescore* allowed for saving under. Plans for the way that data samples were going to be created were established. The music sheets found in *Musescore* were spliced into three different files and then a script would be created that would bring together the spliced music sheets. These all would come together to form a data sheet. Training was conducted to be able to navigate the *Musescore* application and to fully understand the scope as to why this was important to be very careful with each music sheet. A general introduction to music sheets and the notation was conducted so a better understanding was gained from all. Each music sheet has several layers in which are important to the overall design of the web application. The data samples must be formulated accurately for it to fit into what is needed by the software.

5. OMR Development

Developing a network that would implement OMR was delayed until after data collection proceeded to a later stage. Data samples are needed to begin developing the framework for the application for testing to later ensue.

References:

<https://musescore.org/en>

<https://github.com/>

<https://www.lucidchart.com/>

Report #2 Group #1 – 2/10/2020 – 2/16/2020

1. Web Application Development

A request was sent to a staff member at Kennesaw State University to obtain a school server for the group to use. The status of this server and whether we will obtain one is still unknown. Django is a Python web framework and was needed for this type of application. An issue arose as many were not aware on how to use this new framework. Learning how to navigate through Django turned into a weeklong task as developers used a learn as you go approach. Research and planning were conducted so that developers were able to start writing out the application methods.

2. Data Collection

Musecore is the application used to create the data samples. The *Musecore* website was used to download music sheets. The music sheets were searched by using band instruments. Band instruments range from a diverse group of instruments, but an example of one would be a cello or a violin. Now, there was another section of the music sheet that was focused on. The number of bars in the music sheets were looked at as the focus was only one 1-bar and 2-bar music sheets. *Musecore* has a daily limit of only downloading 20 sheets per day which proved to be a struggle at times when diversity of samples was sought after. For downloading purposes, these sheets were downloaded to the filetype: Musecore type. This served a purpose because it must be accessed through the *Musecore* desktop application. The music knowledge gained served its purpose in creating the data samples. Each sample will take the following steps to be created:

1) Format the music sheet so that only a section is being observed and edited

2) Save this file as xml file as well as a pdf file

* .xml would be named 1

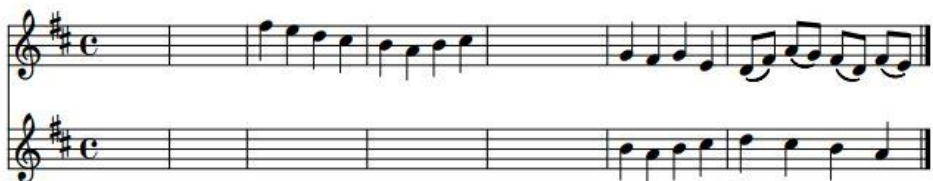
* .pdf would be named 1a

3) Choose a few different musical elements to make invisible

4) Once the elements are invisible, save as a pdf file

* .pdf would be named 1b

The use of numbers and letters to name the different files served to organize each different file. This will help in the future when the times comes to bring each different file together to create a music sheet. There is a goal to create 500 different samples and the strategy is to get 1-5 different samples from each individual music sheet. The diversity of samples was needed to train the software to learn the different handwritten samples and check for multiple student errors.



3. OMR Development

We plan to continue working on the OMR Development next week.

Report #3 Group #1 – 2/17/2020 – 3/1/2020

Web Application Development: So far, we have only developed part of the *student* account. Below are some screenshots of some of the webpages from the sign in page to the *student* dashboard. We plan to start working on the instructor side of the application, and the other data connections this following week.

Welcome Page



WELCOME

"Music gives a soul to the universe, wings to the mind, flight to the imagination and life to everything."
— Plato

Login:

A screenshot of the login page. The background is a dark grey version of the 'Welcome' page. Overlaid in the center is a white login form. The form has two input fields: 'Username' with a placeholder 'Enter Username' and 'Password' with a placeholder 'Enter Password'. Below the password field is a black 'Login' button. Underneath the button is a checkbox labeled 'Remember me' which is checked. At the bottom of the form are two links: a red 'Cancel' button and a blue 'Forgot password?' link. On the right side of the page, there is a dark vertical sidebar with a close 'X' icon at the top and a list of links: 'Login', 'Create Account', 'Plagiarism', 'Resources', 'Support', and 'Contact Us'.

Home:



Good evening Tadiwa!



Class Portfolio My Grades MoreStuff MoreStuff

Sunday
22:47:57

Assignment	Completion Status	Score	Evaluation Status	Due Date
[Section Name]				
Assignment 1	2 Submissions, 2 Files	95 / 100 - 95 %	Feedback: Read	Feb 14, 2020 11:59 PM
Assignment 2	No Submissions	--	--	Feb 28, 2020 11:59 PM



Assignments:



Class Portfolio **My Grades** MoreStuff MoreStuff

Grade Item	Points	Grade	Feedback
Assignments		95%	
Assignment 1	9.5 / 10	95 %	3. -5, not the same as the BNF rule due to the parentheses.
Assignment 2	-- / 10	-- %	--
Quizzes		90%	
Q 1	9.5 / 10	95 %	good job
Q 2	8.5 / 10	85 %	see me

Data Collection:

Now that the music sheets were edited and the musical elements were removed, the next process is to combine the different pieces of the music sheet into one. These pieces would create one data sample. A script was created in python to be able to combine the

different files of one music sheet to create a data sample. The data sample is what students will use to write in the missing musical elements that were removed in the previous week. Upon noticing that there were a few files done incorrectly, these were removed from the data selection. Further assistance is needed from Dr. Yunek to begin collecting data from his music theory students. The application needs student input to hand write in answers to the data sample sheets that we have created. These data sample sheets are important to be overall software development because it will train the machine to accurately pinpoint student errors. These data sample sheets will then be applied against an answer key that has been provided by Dr. Yunek. These data sample sheets are important to the overall software development because it will train the program to accurately pinpoint student errors and provide a marked-up image.

Sample 1: Example of a filled out assignment

WS 21: Interval Construction I

Name Jane Doe

A. Construct the following Major (M) and Perfect (P) intervals above the given pitch. Notes lacking an accidental are understood as natural; **do not change them in any way.**

Interval Construction I

Staff 1 (Treble Clef):

- P4: C4 to F4
- M3: C4 to E4
- P5: C4 to G4
- M6: C4 to A4
- M2: C4 to D4
- M7: C4 to B4
- P5: C4 to G4
- M7: C4 to B4

Staff 2 (Bass Clef):

- M6: C3 to A3
- P5: C3 to G3
- M3: C3 to E3
- M7: C3 to B3
- M2: C3 to D3
- P4: C3 to F3
- P5: C3 to G3
- M6: C3 to A3

Staff 3 (Alto Clef):

- M3: C4 to E4
- P4: C4 to F4
- M6: C4 to A4
- M3: C4 to E4
- M7: C4 to B4
- P4: C4 to F4
- M2: C4 to D4
- M6: C4 to A4

Sample 2: Maximum points assigned to each field to determine grade

WS 21: Interval Construction I

Name Jane Doe

A. Construct the following Major (M) and Perfect (P) intervals above the given pitch. Notes lacking an accidental are understood as natural; **do not change them in any way.**

Interval construction exercises on three staves (Treble, Bass, and Bass). Each staff shows a sequence of intervals constructed above a given pitch. The intervals are labeled below the staves: P4, M3, P5, M6, M2, M7, P5, M7 (Treble); M6, P5, M3, M7, M2, P4, P5, M6 (Bass); M3, P4, M6, M3, M7, P4, M2, M6 (Bass). Each interval is represented by a note on a staff with a blue bounding box around it.

points=8

points=8

points=8

Sample 3: Bounding boxes over incorrect answers causing a lower total score

WS 21: Interval Construction I

Name Jane Doe

A. Construct the following Major (M) and Perfect (P) intervals above the given pitch. Notes lacking an accidental are understood as natural; **do not change them in any way.**

Interval construction exercises on three staves (Treble, Bass, and Bass). Each staff shows a sequence of intervals constructed above a given pitch. The intervals are labeled below the staves: P4, M3, P5, M6, M2, M7, P5, M7 (Treble); M6, P5, M3, M7, M2, P4, P5, M6 (Bass); M3, P4, M6, M3, M7, P4, M2, M6 (Bass). Each interval is represented by a note on a staff with a red bounding box around it.

6/8

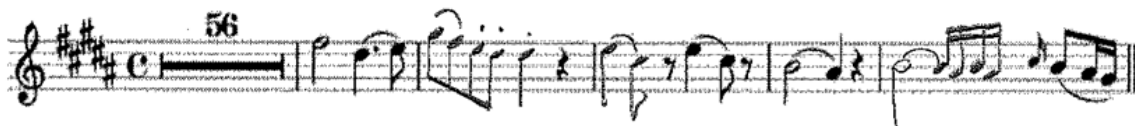
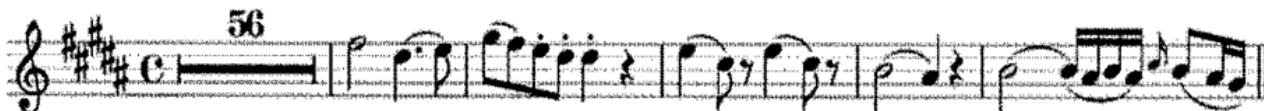
8/8

8/8

Sample 4: Data collection form to obtain samples of different handwritings from students.



Sample 5: Filled in data collection form to obtain samples of different handwritings from students.



Data Storage:

Creation of the Autograder database to store information and send data from the OMR to the web application. MariaDB composed of 11 tables inserted by command line.

OMR Development:

Researching the different classes of notes/rests needed to classify objects required to train a CNN using [1][2] as references. The main concerns of the CNN are the number of objects needed to be recognized on a single image where notes/rests have the most variance in types, durations, and pitch.














Sharp	Flat	Natural	G clef	F clef	C clef	Bar Line	Time Sig
#	b	♮					
Notes				Rests			
							

Fig 1. [2] depiction of the 10 classes of symbols with 9 types of notes/rests needed for object detection



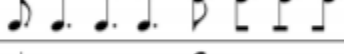




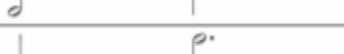
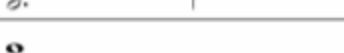

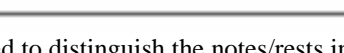

Duration (♩=1)	Label	Notes	Rests
0.125	1		
0.250	2		
0.500	3		
0.750	4		
1.000	5		
1.500	6		
2.000	7		
3.000	8		
4.000	9		

Fig 2. [2] depiction on duration needed to be encoded to distinguish the notes/rests into discrete categories of the same class.



Fig 3. [2] depiction on pitch that is encoded based on vertical distance

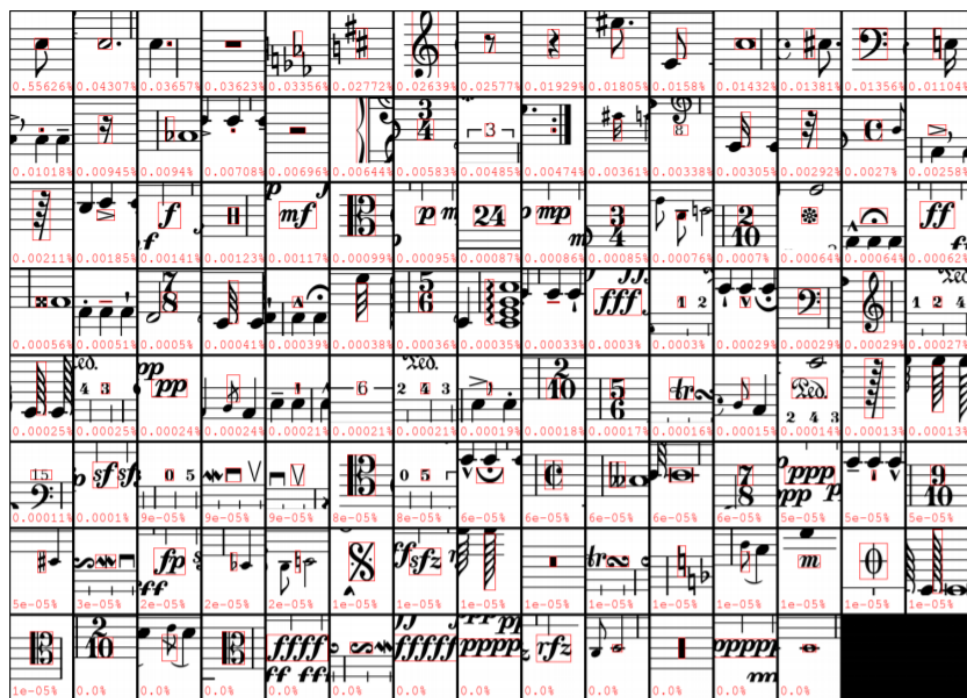


Fig 4. Key detection objects that will account for most objects and will allow the creation of bounding boxes to be parsed by the SVG file as shown in [1].

References:

- [1] https://tuggeluk.github.io/papers/preprint_deepscores.pdf
- [2] [State-of-the-Art Model for Music Object Recognition with Deep Learning](#)

Report #4 Group #1 – 3/2/2020 – 3/22/2020

1. Web Application Development

Application functions completed during this time were (1) account creation and login using connections to the server-side database for both *student* and *instructor* type accounts, (2) class, instructor, student, and assignment relations connected to the database. Due to issues using Python *django* for framework for the application back end, we switched to a PHP application back end except as relates to the OMR development which remains under construction in Python. Because of the switch, we had to redesign and improve the entire site and the database relations. We have made them more effective and efficient, in a way that allows us to add and pull out functions as separate modules, without affecting other parts of the application. So far, the entire database has been created but we continue to add new elements to it as we see fit. We have also changed the site map to be simpler and easier for all users. The Sign In and Account Creation pages will look as shown below.

Sign in:



Email

instructor@chordscore.com

Password

....

By signing in, you agree to our [Terms & Privacy](#).

Sign in

[Forgot Password?](#)

New? [Sign Up](#).

Copyright © ChordScore 2020

Create Account:



Create Account

Email

Enter Email

Password

Enter Password

Repeat Password

Repeat Password

Choose Account Type: Select

By signing in, you agree to our [Terms & Privacy](#).


Sign up


Already have an account? [Sign In](#).

Copyright © ChordScore 2020

We have also improved the *instructor* and *student* accounts to have identical features, as shown below, for simplicity and uniform design purposes. Refer to the images below.

Instructor:

Home Settings Signout

TADIWA MANGADZE
instructor@chordscore.com

Mon, Mar 23, 2020

COURSE NAME & SECTION

Announcements

All Classes

All Students

Assignments

Create New Assignment

Post Announcement

Messages

Signin History

Logout

Announcements


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas ultricies eget ante vulputate tempus. Ut vulputate nisi sed lorem posuere mollis. Suspendisse congue tempor arcu, at lacinia lorem sollicitudin eget. Nullam sit amet hendrerit nisi. Suspendisse feugiat ut tellus quis aliquam. Mauris euismod justo quis volutpat accumsan. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum ut hendrerit lorem.


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas ultricies eget ante vulputate tempus. Ut vulputate nisi sed lorem posuere mollis. Suspendisse congue tempor arcu, at lacinia lorem sollicitudin eget. Nullam sit amet hendrerit nisi. Suspendisse feugiat ut tellus quis aliquam. Mauris euismod justo quis volutpat accumsan. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum ut hendrerit lorem.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas ultricies eget ante vulputate tempus. Ut vulputate nisi sed lorem posuere mollis. Suspendisse congue tempor arcu, at lacinia lorem sollicitudin eget. Nullam sit amet hendrerit nisi. Suspendisse feugiat ut tellus quis aliquam. Mauris euismod justo quis volutpat accumsan. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum ut hendrerit lorem.

Copyright © ChordScore 2020

Student:

Home Settings Signout

AUSTIN ROGERS
student@chordscore.com

Mon, Mar 23, 2020

COURSE NAME & SECTION

Announcements

Assignments

Grades

Submit Assignment

Resource

Messages

Signin History

Logout

Announcements

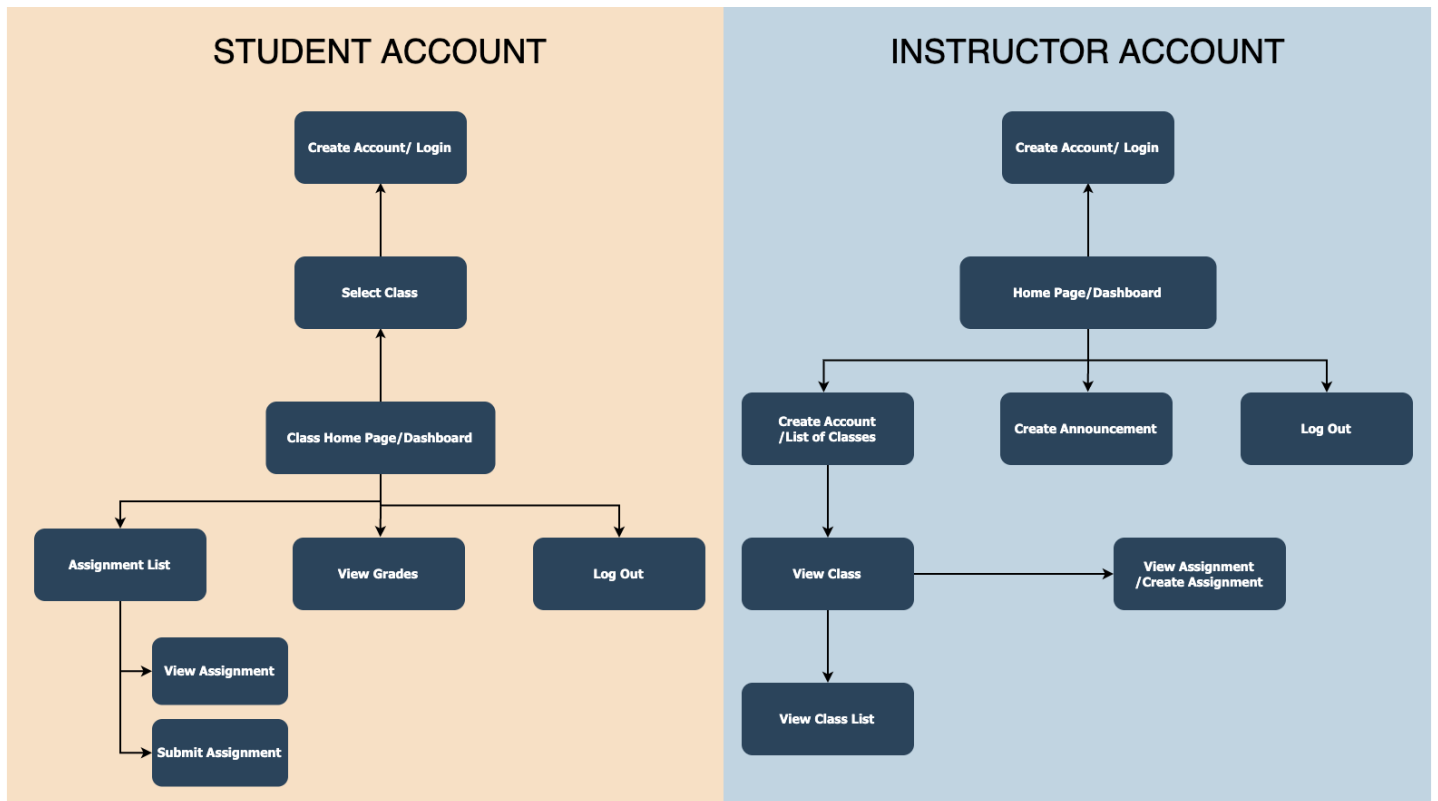
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas ultricies eget ante vulputate tempus. Ut vulputate nisi sed lorem posuere mollis. Suspendisse congue tempor arcu, at lacinia lorem sollicitudin eget. Nullam sit amet hendrerit nisi. Suspendisse feugiat ut tellus quis aliquam. Mauris euismod justo quis volutpat accumsan. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum ut hendrerit lorem.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas ultricies eget ante vulputate tempus. Ut vulputate nisi sed lorem posuere mollis. Suspendisse congue tempor arcu, at lacinia lorem sollicitudin eget. Nullam sit amet hendrerit nisi. Suspendisse feugiat ut tellus quis aliquam. Mauris euismod justo quis volutpat accumsan. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum ut hendrerit lorem.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas ultricies eget ante vulputate tempus. Ut vulputate nisi sed lorem posuere mollis. Suspendisse congue tempor arcu, at lacinia lorem sollicitudin eget. Nullam sit amet hendrerit nisi. Suspendisse feugiat ut tellus quis aliquam. Mauris euismod justo quis volutpat accumsan. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum ut hendrerit lorem.

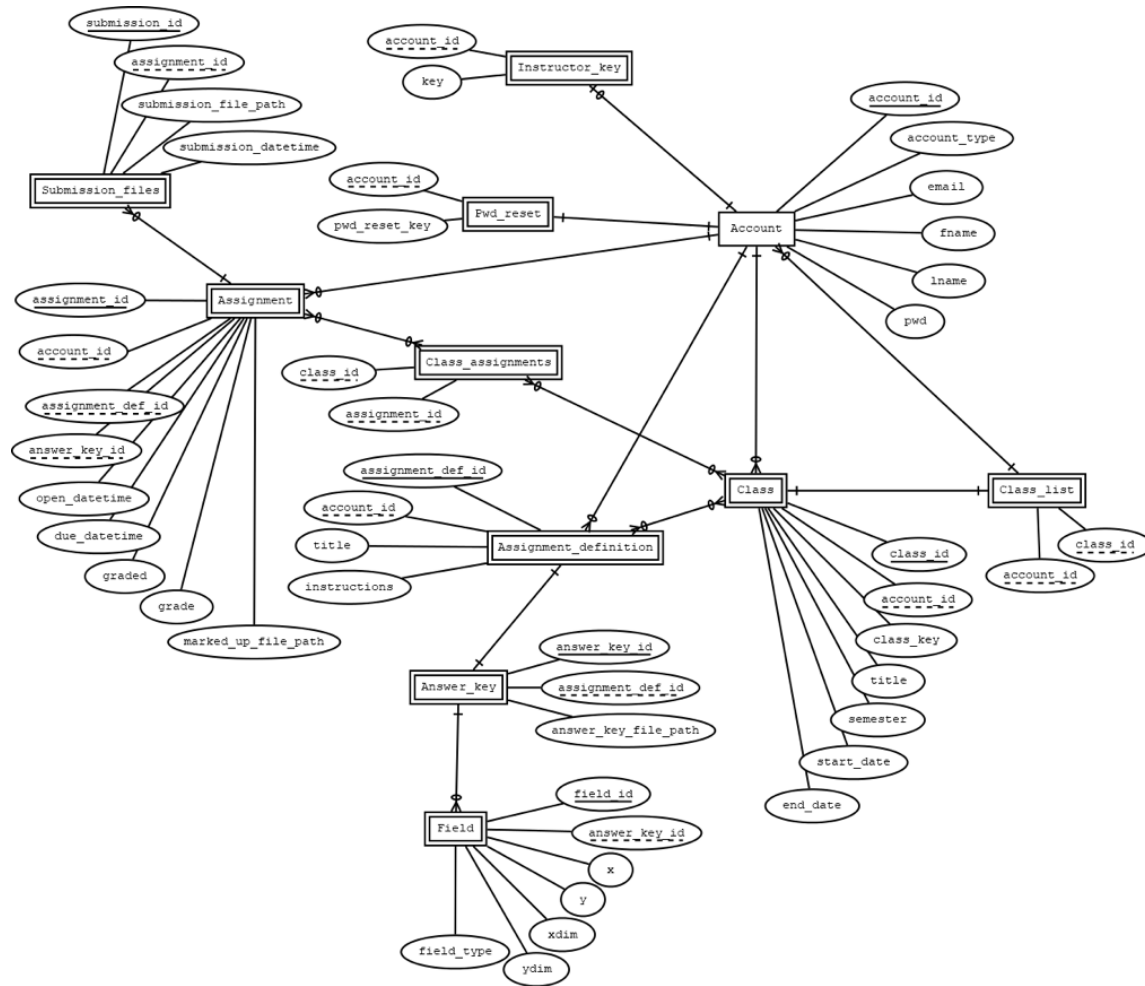
Copyright © ChordScore 2020

The site map of essential elements of the application will look like the diagram below:



1.1 ER Diagram

Created an ER diagram to show the relationships between the different tables in our Autograder database. This shows the how the relationships of the entities are stored into the database. This helps us to visually note what changes are needed to get done without directly affecting the database itself. It becomes a tricky tactic to make changes directly into the database, so this diagram serves a purpose of making sure beforehand what inter-relationships are needed and how they work. This will help substantially to create a functional database.



2. Data Collection

Plans to collect samples of music notation with a mixture of handwritten and printed notes from students for training a notation recognition model had to be re-invented with the campus closure on 3/16/2020. Instead of gathering samples with mixed printed/handwritten notation, we decided to train the model using two different datasets to train it – one completely printed (the *Deepscores* dataset [1]) and one completely handwritten (the *MUSCIMA++* [2]).

2.1 Data Format Requirements

Based on the paper after which our methods are modeled, the task of recognizing musical objects on a staff and converting them to a computer-readable, digital format (OMR) should be broken into two steps before combination into an end-to-end network can be implemented. These are *Object Identification & Classification* and *Digital Format Generation*. These steps, along with network architecture will be described in more depth below, but as regards data collection, these steps implicate the need for music notation (whether printed or handwritten) in three different forms...

- Music page image of proportions equivalent to the standard 8 ½ x 11 page
- The set of bounding boxes for each classifiable object on the page (box location & dimensions + object classification)
- A digital music notation from which the following information can be extracted or derived for each *note*, *rest*, *clef*, *key-signature*, and *time-signature*
 - Pitch (applies to *note* and *key-signature*)
 - Duration (applies to *note* and *rest*)
 - Voice (applies to *note* and *rest*)
 - Class (applies to *note*, *rest*, *clef*, *key-signature*, and *time-signature*)
 - Accidentals (applies to *note*)
 - Chord (Boolean True if part of a chord, False otherwise; applies to *note*)
 - Time modifications (applies to *note* and *rest*)

2.2 Data Sources Based on Requirements

Modeled after the paper in [3], we attempted to generate all three music notation forms (2.2.a-c) from downloaded MUSICXML files using the MuseScore application's command-line interface. The MuseScore CLI allows conversion of any MUSICXML file to both Portable Network Graphics (PNG) and Scalable Vector Graphics (SVG) file formats. The PNG format satisfied requirement 2.2.a. MUSICXML files satisfy requirement 2.2.c. To achieve requirement 2.2.b by providing bounding boxes for all identifiable musical elements, we attempted to parse the SVG file format for the bounds around musical elements. Below is a sample of SVG file content and the corresponding images before (left) and after (right) red boxes were inserted around the bounds of a "notehead" element based on the minimum and maximum x and y coordinates used to draw the element in the SVG file:



We could not determine a way to derive the same bounding boxes for handwritten student data samples which would prove to be an issue for an end-to-end neural network. In response to this, we decided to use instead the *Deepscores* and *Muscima++* datasets ([1] and [2]) for printed and handwritten data respectively. As far as we are aware, both datasets can provide all three-music notation forms we require. We are currently determining the best way to obtain each format, convert it to a standard format, and encode it for use in creating our model.

```
PS C:\Users\Ruth\Documents\KSU\ChordScore\musicxml> MuseScore3 -o lg-108945789.png lg-108945789.xml
PS C:\Users\Ruth\Documents\KSU\ChordScore\musicxml> MuseScore3 -o lg-108945789.svg lg-108945789.xml
PS C:\Users\Ruth\Documents\KSU\ChordScore\musicxml> ls
```

Directory: C:\Users\Ruth\Documents\KSU\ChordScore\musicxml

Mode	LastWriteTime	Length	Name
-a----	3/22/2020 9:11 PM	87513	lg-108945789-1.png
-a----	3/22/2020 9:11 PM	63707	lg-108945789-1.svg
-a----	3/3/2020 4:27 PM	18411	lg-108945789.xml

3. OMR Development:

The neural network design and architecture are still under way. The input layer to the network must be a Convolutional Layer that takes pixels of the music notation images as input. The network must then be trained to predict output in two stages. In the first stage, the network predicts the bounding boxes of relevant musical elements using the input images and bounding box data for training. The second stage takes the second to last layer from the first network as input, and outputs a sequence of measures (each measure containing a sequence of notes with attributes as defined in the third music notation format requirement). The following diagrams model our base network architecture. Fig 3.1 describes the first stage of the network (object detection through bounding box prediction) and Fig. 3.2 describes the second stage (music reconstruction in the digital for as a series of measures).

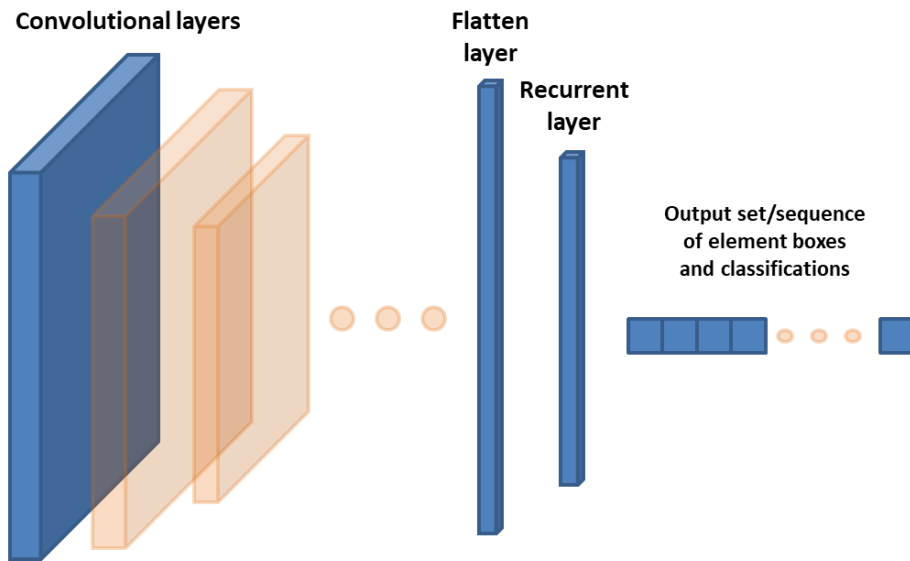


Figure 3.1 Network for first stage – object detection via bounding box prediction

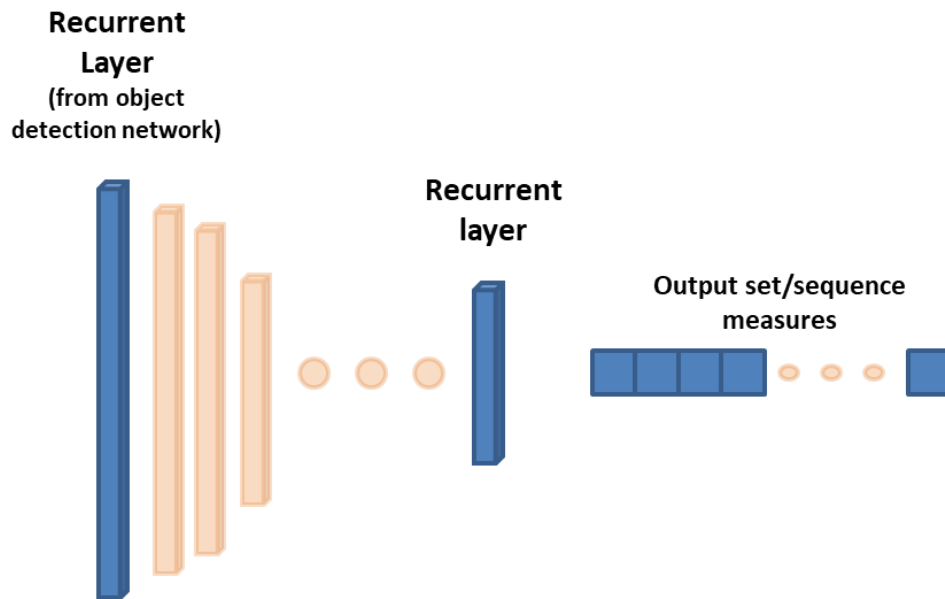


Figure 3.2 Network for second stage – measure sequence generation

References:

- [1] <https://tuggeluk.github.io/deepscores/>
- [2] <https://ufal.mff.cuni.cz/muscima>
- [3] [State-of-the-Art Model for Music Object Recognition with Deep Learning](#)

1. Web Application Development

The web application interface and web page design has been completed except for the Instructor-side Assignment Creation page which will require significantly more JavaScript code as it will allow instructors to drag and drop boxes to define fields on the assignment page where the automatic grader is to expect student responses.

Web application backend is nearly complete. We have defined a set of features for a minimally viable product on the web application side. All of these features have been implemented (with connection to the server database) except for the *Create a new assignment*, *Edit an existing assignment*, and *Import saved assignment definition* features. These are separated into features for students (left) and instructors (right). (See Fig. 1)

2. Data Collection

We are preparing data from our two datasets of choice *Deepscores* (images of printed music) and *MUSCIMA++* which for the most part supply the data

formats we require. Data from these datasets must be in three formats – raw images, object bounding box arrays, and sequences of chord objects that capture the sequential and semantic structure of the music. As soon as data preparation is finished, we will build a C-RNN network that is trained by (1) using *raw images* to predict *object bounding box arrays* and (2) using this first network to predict *object sequences*. The three data formats are specified below.

2.1. Raw Image format

Desired format: Each image sample must be a single page JPEG image file with an aspect ratio equivalent to a standard sheet of paper (8.5 x 11). This is required to use these images as input to a Convolutional ANN layer. The image will be resized to a size determined experimentally to best optimize the neural network. Our initial tests will use images resized (aspect ratio maintained) to a width of 600 pixels as seems the minimal size reduction that does not lose significant image details to the human eye.

Deepscores [1] images do not require preprocessing beyond resizing and conversion from PNG to JPEG files, and they are separated into single-page samples such that each page in a multipage piece is treated as a separate data sample. One image maps to one object bounding box file, and the corresponding *musicxml* file can be parsed and segmented into one corresponding page. See the image right as a sample image.

MUSCIMA++ [2] tools (related to the *CVC-MUSCIMA* project [3]) images also do not require preprocessing beyond image resizing. This dataset contains handwritten scores that were copied from 20 distinct scores. Each score was transcribed by 50 different

musicians creating a total of 1000 handwritten scores. This dataset was originally intended for music transcriber identification rather than OMR, but we attempt to repurpose



Figure 2 - Raw Image from CVC-MUSCIMA dataset

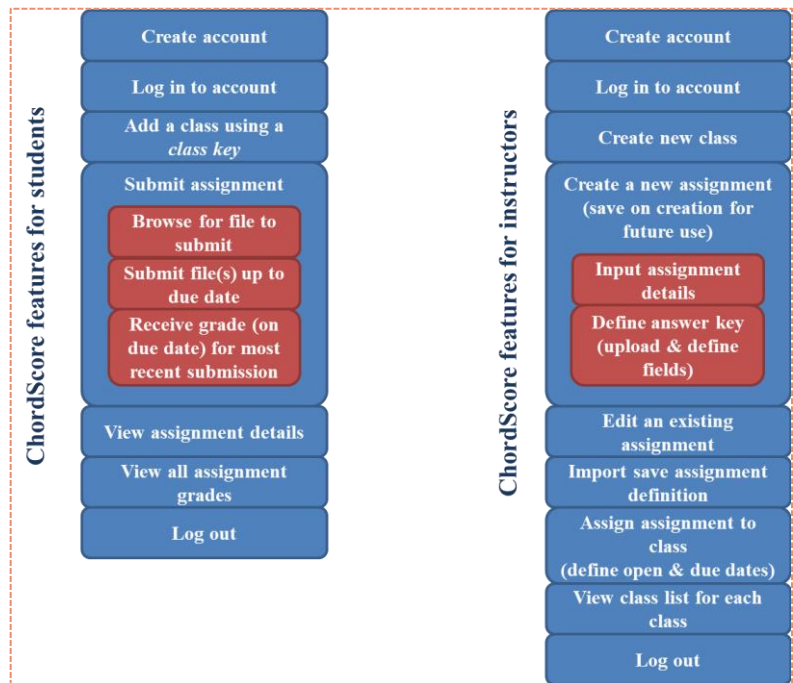


Figure 1 - Feature Specification



Figure 3 - Raw Image from Deepscores dataset

it due to our inability to collect our own handwritten samples. See Fig. 3.

2.2. Object bounding box array format

Bounding box arrays will be a sequence of bounding box objects with data fields for the *minimum x-coordinate*, *minimum y-coordinate*, *maximum x-coordinate*, *maximum y-coordinate* of the bounding boxes and *class name* of each object. Bounding box coordinates will be mapped to the corresponding coordinates on the resized images. The bounding box data will be converted to an $N \times 41$ array, where N is the number of bounded objects in a samples and 41 is the 4 box coordinates + a one-hot-encoded array of 37 object classes. Both the *Deepscores* and *CVC-MUSCIMA* datasets label bounding boxes with a very thorough symbol classification scheme required for OMR in full scores. However, the scope of the ChordScore project is limited to student homework. Only the following object classes are necessary for determining the correctness of a student homework submission for a total of 37 object classes:

<i>note</i>	<i>dotted-note</i>	<i>rest</i>	<i>dotted-rest</i>	<i>semantic symbols</i>	<i>accidentals</i>
whole-note	dotted-whole-note	whole-rest	dotted-whole-rest	treble-clef	natural
half-note	dotted-half-note	half-rest	dotted-half-rest	bass-clef	flat
quarter-note	dotted-quarter-note	quarter-rest	dotted-quarter-rest	tenor-clef	double-flat
8th-note	dotted-8th-note	8th-rest	dotted-8th-rest	time-signature	sharp
16th-note	dotted-16th-note	16th-rest	dotted-16th-rest		double-sharp
32th-note	dotted-32th-note	32th-rest	dotted-32th-rest		
64th-note	dotted-64th-note	64th-rest	dotted-64th-rest		

Deepscores and *CVC-MUSCIMA* both offer bounding box object data in XML format. We are writing scripts to parse files from each database to convert them to the $N \times 41$ array format. The resulting arrays will contain only the objects from one of the necessary 37 classes. Figs. 4 and 5 give examples.

Converting these bounding boxes to array format will require some engineering since both *CVC-MUSCIMA* and *Deepscores* separate object classes at a more integral level than we need. For example, for ChordScore,

half-note
= **noteheadBlack+noteStem**
from *Deepscores* and
half-note
= **notehead-full+stem**
from *CVC MUSCIMA*.

2.3. Chord sequence format

Chords in a chord sequence are matrices containing encoded information for each relevant object from music notation.

Chord sequences capture both

the note values and the semantic structure of music. For example, an instructor might specify that notes *B, B, C, G* are the correct answer to a problem. If a student were to answer *B, B, G, C*, the student would have written the correct notes, but the last two notes, *C* and *G*, are incorrect because the order is wrong. Chord sequences preserve the music structure as a sequence of notes, some of which are stacked into chords. Our desired chord sequence format is a sequence of arrays of size $4 + 350 (= 354)$. The first 4 values are *measure number*, *staff number*, *voice number*, and *duration*. The remaining 350 values are one-hot-encoded note pitches and accidentals – 70 possible “raw” pitches (no accidentals) and 5 possible accidentals. We are still trying to figure out how to handle clef and key signature objects which are important but are not chords.

```
<?xml version="1.0"?>
- <annotation>
  <folder>Annotations</folder>
  <filename>lg-2267728-aug-beethoven--page-2.svg</filename>
  - <size>
    <width>2707</width>
    <height>3828</height>
    <depth>1</depth>
  </size>
  <segmented>0</segmented>
  - <object>
    <name>noteheadBlack</name>
    - <bndbox>
      <xmin>0.66612573</xmin>
      <xmax>0.67682069</xmax>
      <ymin>0.10420239</ymin>
      <ymax>0.11078642</ymax>
    </bndbox>
  </object>
  - <object>
    <name>noteheadBlack</name>
    - <bndbox>
      <xmin>0.53657921</xmin>
      <xmax>0.54727418</xmax>
      <ymin>0.13402136</ymin>
      <ymax>0.14060538</ymax>
    </bndbox>
  </object>
</annotation>
```

Figure 4 - Bounding box data from *Deepscores*

```
<?xml version="1.0" encoding="UTF-8"?>
- <CropObjectList xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance">
  - <CropObjects>
    - <CropObject xml:id="MUSCIMA-pp_1.0__CVC1">
      <Id>0</Id>
      <ClassName>notehead-full</ClassName>
      <Top>372</Top>
      <Left>494</Left>
      <Width>29</Width>
      <Height>20</Height>
      <Mask>0:15 1:10 0:14 1:16 0:11 1:20 0:7</Mask>
      <Outlinks>730 575 771 797</Outlinks>
    </CropObject>
    - <CropObject xml:id="MUSCIMA-pp_1.0__CVC2">
      <Id>1</Id>
      <ClassName>notehead-full</ClassName>
      <Top>373</Top>
      <Left>557</Left>
      <Width>24</Width>
      <Height>18</Height>
      <Mask>0:18 1:3 0:17 1:8 0:11 1:13 0:11 1:13</Mask>
      <Outlinks>241 382 535 771 797</Outlinks>
    </CropObject>
  </CropObjects>
</CropObjectList>
```

Figure 5 - Bounding box data from *CVC-MUSCIMA*

For the *Deepscores* dataset, we are working to parse these sequences from *musicxml* files. The *CVC-MUSCIMA* dataset does *not* provide a ground truth digital music format such as *musicxml*. We will remedy this by inputting the 20 distinct, one-page scores from this dataset into a *musicxml* file with the MuseScore application which we consider reasonable effort for a 1000-sample database.

3. OMR Development

Development of the C-RNN for OMR is on hold until data preparation has been implemented on enough samples to create a test run (or dummy run).

4. References

- [1] <https://tuggeluk.github.io/deepscores/>
- [2] <https://ufal.mff.cuni.cz/muscima>
- [3] http://www.cvc.uab.es/cvcmuscima/index_database.html

Project Schedule

Day/Week/ Module	Tasks
Week 1 (01/06 - 01/12)	Project Discussion and Task Dividing
Week 2 (01/13 - 01/19)	Project Outline
Week 3 (01/20 - 01/26)	Creating a framework and the schedule
Week 4 (01/27 - 02/02)	Requirements Documents Due Data Format Requirements created and communicated to Dr. Yunek
Week 5 (02/03 - 02/09)	Application Development
Week 6 (02/10 - 02/16)	Obtaining Data Samples
Week 7 (02/17 - 02/23)	Design Revision
Week 8 (02/24 - 03/01)	Application Framework Due (Front end & Back end implemented at minimum functionality and working together)
Week 9 (03/02 - 03/08)	Application testing and debugging
Week 10 (03/09 - 03/15)	Data Collection
Week 11 (03/16 - 03/22)	Document Revision, finalizing application development
Week 12 (03/23 - 03/29)	Last minute revisions

Week 13 (03/30 - 04/05)	Machine Learning Tests
Week 14 (04/06 - 04/12)	Application Working and Tested
Week (04/13 - 04/19)	Final Project Report Due
Week 15 (04/20 - 04/26)	Presentation Hand off project to Dr. Yunek
Week 16 (04/27 - 04/30)	Presentation