1. Intro
   1. Goal
      1. Develop a sheet music recognition system for scanned/electronic music scores as good as possible within the time limit
      2. Output: MusicXML file
   2. Motivation
      1. Learning tool
      2. Composition – to have a baseline and rearrange for other instruments
      3. Enjoyment – what an arrangement sounds like
      4. Personal connection – manually inputting scores
   3. Users
      1. Instrument learners
      2. Composers
   4. Software
      1. Java program
      2. User interface – progress displayed
   5. Limitations
      1. No proper binarization for scanned scores
      2. Assuming 300 dpi clear image
      3. No hand-written scores/old-fashioned notations
      4. Only some music notation
2. Whole project (Entertainer)
   1. Deskewing
      1. SGD – rotate by a small angle until error increasing
      2. Histogram
      3. Function – threshold, add values
      4. Switch direction
      5. Rotation – library image transformation
   2. Stave removal
      1. Histogram, threshold
      2. Find bottom
      3. Don’t remove if something above/below
      4. Remove extra pixels due to distortion
      5. Record average stave space as a measure
   3. Vertical line removal
      1. Find black pixel
      2. Find width (assume some minimum width)
      3. Find height, allow x pixels to stick out and a gap of y pixels
      4. Find width again to the left and right, at least a fraction of the height of pixels in a column
      5. It’s a line if width < x and height > y
      6. Bar line recognition
         1. Beginning and end of stave
         2. Not bar line if connected to something thick enough – note head, but might be connected to a tie/slur
      7. Remove line
         1. Don’t remove if something to the left or right
         2. Remove extra pixels due to distortion
         3. Only remove if x consecutive pixels to be removed – don’t break up crotchet rests – unless the line is a stem
   4. Patching
      1. Because of broken symbols, especially minims and semibreves
      2. Find two paths between stave lines
      3. Check if curved/diagonal
      4. Check if no straight line present, unless stem
   5. Connected component analysis
      1. Scan image with a mask
      2. Assign labels
      3. Record label equivalence
      4. Replace labels with their representative label
      5. Draw bounding boxes
   6. Symbol recognition
      1. Discard noise (1x1 components)
      2. Discard anything out of bounds
      3. Decision tree
      4. Initial recognition (beams, dots, tails, ties, slurs)
      5. Further recognition (notes, accidentals, rests)
         1. Loop through each system
         2. Loop through each measure
         3. Loop through each stave
         4. Collect and sort by x components within bounds
      6. For notes and rests, look for modifying symbols (dot, beam, tail, tie, slur)
      7. Record a note
         1. Determine pitch based on vertical position
         2. Put into a chord if multiple notes with the same x position
         3. Discard as noise if very close to the previous note but different duration
      8. Convert everything into MusicXML
3. Project management
   1. Logbook of ideas (including rejected ones)
   2. Literature review before every stage
   3. Git repository – single features after testing
   4. Systematic work every week
   5. Skip difficult stages to develop a complete system
   6. If more time, would do:
      1. More complex notation – chords, dynamics, glissandos
      2. Binarization, better rotation (shearing) and line removal (Hough transform)
      3. Projections
   7. If again, would’ve focussed on developing the first prototype more quickly and then worked on whatever needed most improvement