Project Proposal Components [15 pts]
Write up a proposal file (in the file proposal.txt, or .docx, or .pdf) which should include the following components:

- **Project Description** [2.5 pts]: The name of the term project and a short description of what it will be.
    - Name: Kumon Simulator
    - This project is essentially a kumon simulator. First, the user will click on a menu to select the type of math problem they desire. The computer will then generate a problem and the user will need to write the answer in a box with their cursor, which the computer will double check.
- **Similar projects** [2.5 pts]: A 1-2 paragraph analysis of similar projects you've seen online, and how your project will be similar or different to those.
    - After a very brief look on the internet, I found some websites with similar aspects with my project. But the websites I found are only half my project. For example, Khan Academy and Web Math Minute generate problems for the user to solve without a teacher to help, but the user will need to type(not write) the answer. Meanwhile, websites such as Texthelp enable students to turn their writing into text/equations, but can not generate problems and check the students answers (need a human teacher to do that).
    - I think a deeper look might allow me to find more similar projects.

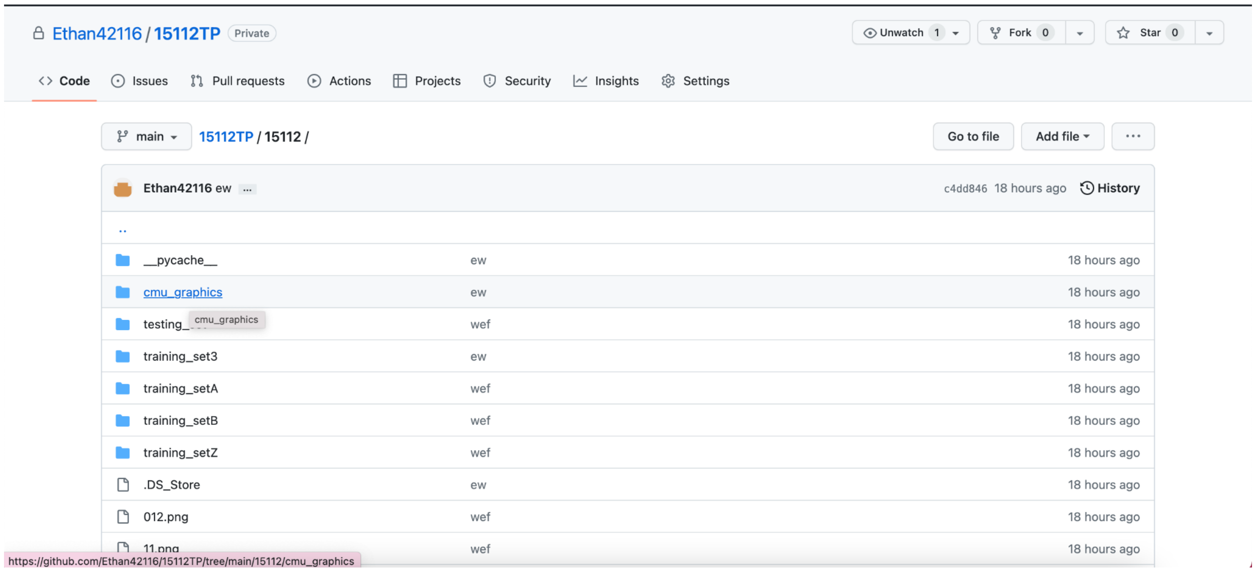
- **Structural Plan** [2.5 pts]: A structural plan for how the finalized project will be organized in different functions, files and/or classes.
    - Front end file: generated user interface with CS3
    - Image handler file: a file that contains everything that is related to PIL
        - Generating training data/images for the AI to read handwriting
        - Converting the user's handwriting into a image
        - Breaking the user's handwriting into individual letters
    - AI file: use sckitlearn's AI to read the individual letters to determine what the user is writing
    - Equation/problem generator file (might be a class)
        - Helps generate and solve a math problem based on various inputs
    - File navigator function
        - A function that recursively searches through folders for it's contents
        - Is needed to quickly train AI
- **Algorithmic Plan** [2.5 pts]: A plan for how you will approach the trickiest part of the project. Be sure to clearly highlight which part(s) of your project are algorithmically most difficult, and include some details of how you expect to implement these features.
    - Randomly remove some pixels from a training image to distort it (to increase the amount of training mages): works but need some fixes
        - Finds find a n random dark pixels on the image where n is a random number from 1 to 8
        - For each dark pixel, floodfill 1/300 of the total image size around the dark pixel by recursively going to nearby pixels and floodfilling them

    - Splitting the user’s writing into letters: works
        - Loop through the pixels of the center row of the image array
            - If the pixel is dark, floodfill surrounding area with bright pixels and save all pixels that you floodfilled into a set
            - Construct a image from the set of pixels
            - Continue looping
    - Reading the user’s handwriting as a image: had not started yet
        - When the mouseX and mouseY of the user is inside textbox at onMousePress
            - Sets the pixel at mouseX, mouseY to be dark pixel
            - Saves mouseX and mouseY as tuple into a set of dark pixels locations
        - Once finishes writing, convert set of (mouseX,mouseY) to a image

- **Timeline Plan** [2.5 pts]: A timeline for when you intend to complete the major features of the project.
    - Week 0: what I already have
        - A working AI
        - A working AI trainer
        - A working space finder algorithm (through it can not processfractions)
        - Have some very basic (unrelated but important to project) things about training AI and generating training data
    - Week 1:
        - Finish implementing having the AI read the user’s handwriting/having the computer understand what the user is writing
        - Have a very basic equation generator that can generate basic arithmetic problems
    - Week 2:
        - Combine everything into a prototype front end
        - Refine equation generator to output harder problems such as fractions
    - Week 3:
        - Buffer week to catch up

- **Module List** [1 pts]: A list of all external modules/hardware/technologies you are planning to use in your project. Note that any such modules must be approved by a tech demo. If you are not planning to use any additional modules, that's okay, just say so!
    - OS, PIL, Numpy, Emnist, Matplotlib, Scikitlearn, JSON(will very likely be removed because it is doing nothing) are the module’s I am currently using in addition to basic modules such as Random and Copy. All modules are currently installed and working. Additionally, I might add more modules into my project, but I don’t think it would be necessary.

- **Version Control Plan** [1.5 pts]: A short description **and image** demonstrating how you are using version control to back up your code.
    - I am backing my code up on github



15112TP / 15112 /

Ethan42116 ew ... c4dd846 18 hours ago

..		
__pycache__	ew	18 hours ago
cmu_graphics	ew	18 hours ago
testing_	wef	18 hours ago
training_set3	ew	18 hours ago
training_setA	wef	18 hours ago
training_setB	wef	18 hours ago
training_setZ	wef	18 hours ago
.DS_Store	ew	18 hours ago
012.png	wef	18 hours ago
11.png	wef	18 hours ago

https://github.com/Ethan42116/15112TP/tree/main/15112/cmu_graphics

TP1 UPDATE:

- Overall Status: on schedule
- StoryBoard Update: the user writing input textbox will be much wider, and I added an eraser button that allows the user to erase mistakes.
- What I finished this week
    - A textbox for you to write in and for the computer to read. (the biggest and most important feature in project)
        - I made a drawable image CMU graphics using app.draw, fed the image to my space finder function, which then fed everything to the AI
    - A eraser feature
        - When eraser is selected and user drags mouse across his writing, floodfill the current character
    - A basic random problem generator
        - Generates a problem by taking two random numbers and a random operation. Included cases to prevent fractions and division/modulus by 0
    - A very basic front end
        - Combined all above features together and create classes for buttons such the check answer button
- Updated goals before TP2
    - Make menus for me to select problem difficulty
    - Make my equation generator more sophisticated
    - Modify my EMNIST AI to include reading negative sign (70% sure if doable)
    - Make everything look prettier

TP2 UPDATE:

- Overall Status: on schedule, but my prototype front end can look better but I don’t know how
- StoryBoard Update:
  - Added a scoreboard, timer, and a display of difficulty level
  - Removed dropdown menu and self selecting difficulty feature, the app will select difficulty of problem based on your score
  - You can not select type of problem any more
  - Added “click to do another problem” button
  - Completely changed start screen design (ex. Two pages now, no dropdown menu, contains instructions on how to play)
  - Might have more that I missed
- What I finished this week
  - Trained AI to read negative numbers (“-” sign is a M)
    - Generated about 600 images of M using my image generator, labeled them with a “-” sign, and fed them into AI as a 3d array.
  - Successfully implemented fractions
    - Users can now write fractions. The computer will then read the numerator and denominator and divide them.
  - Made equation generator more sophisticated
    - Now have 3 levels of difficulty
    - Level 1 is one operation and no fractions
    - Level 2 is two operations, might have fractions and exponents
    - Level 3 is three operations, might have fractions and exponents
  - Improved front end
    - Updated and polished feature locations and alignment
    - Added a front page and a report issues page
    - Added a scoreboard that determines the problem difficulty the user gets
    - Added a timer that resets after every question and a dashboard that displays the difficulty level of problems
    - Added color palette to make everything look better
  - Improved image generation app
    - Removed front end features that are redundant
    - Made back end function dictating generation more diverse (you can now choose to not modify some features of the image)
    - Fixed randomlyalterV2 back end function to remove different number of pixels for different images (number of dark pixels/9)
  - Since I done a lot this week, might have missed some things
- Updated goals before TP2
  - Make everything look even better
    - Maybe add a gradient for buttons
    - Don’t really know what to do
  - Maybe code my now neural network instead of replying on scikitlearn (not sure how feasible)

TP3 UPDATE:

- Overall status: reached MVP!!
- StoryBoard Update:
  - Added confetti when you get a question correct
- What I finished this week: Not much, since I reached MVP
  - Made check button automatically read an image instead of needing to press k
  - Updated the instructions page front end to make the buttons align to specific instructions about the button
  - Added a confetti animation for when you get something correct
  - Made equation generators unable to come up with negative mods (ex. 1+2%-1)
- Future goals
  - Might have some undiagnosed issues/things to fix in my image generator, need to fix them
  - Made my own neural network instead of relying on sklearn