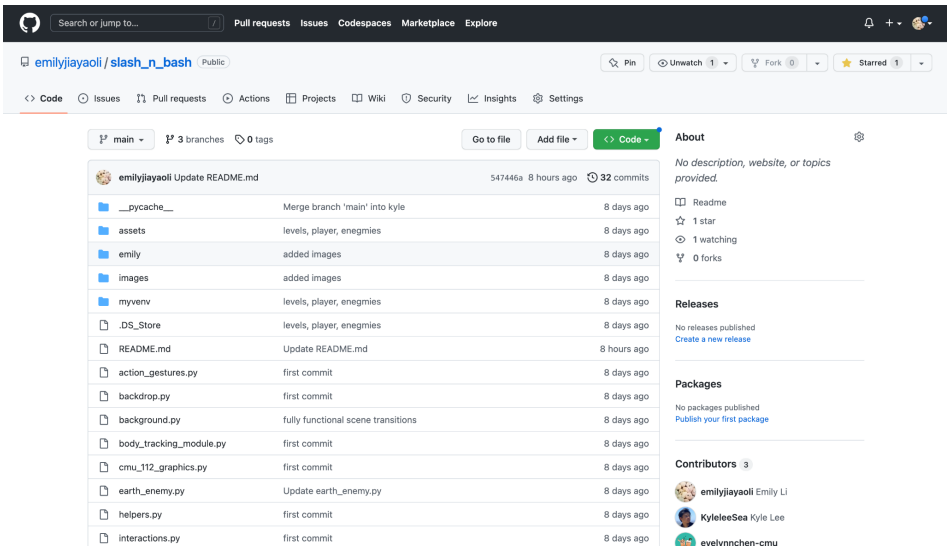
- TP3 Update:
    - Accomplishments
        - Implemented checkmate
        - Implemented en passant
        - Rewrote a portion of code to optimize with OOP, app and chess board interaction
        - Resolved various bugs such as king being able to walk into check
        - Optimization: Utilized more efficient data structures such as sets to avoid unnecessarily iterating through lists
- TP2 Update:
    - Decided to switch project entirely to chessAI
    - Accomplishments: I finished the following features
        - Create board
        - Legal moves
        - Decent UI
        - Castling
        - Pawn Promotion
    - Goal:
        - Fix bug in checkmate
        - Finish implementing en passant
        - Begin to code AI algorithm that plays via minimax/alpha beta pruning and other methods
        - Improve graphics
- TP1 Update:
    - Accomplishments:
        - Create basic piano tiles game with 4, 6, 8 columns
            - made board/grid
            - made tiles show up
            - made tiles move down the screen
            - made recognition based on keypressed
    - New goals and features
        - Two player mode
        - Various levels
        - Progress bar
        - Incorporate audio
    - Challenges I have faced:
        - Determining the data structure used to store the tiles, especially because I will generate random tiles in independent columns
        - Creating a scale for the time at which the tiles are fired. Used counter in timer fired
- Project Description [2.5 pts]: The name of the term project and a short description of what it will be.
    - PianoTilesGo. A multi-level piano tiles game that syncs the player's playing with real time audio with minimal use of external modules. At its advanced stages, this

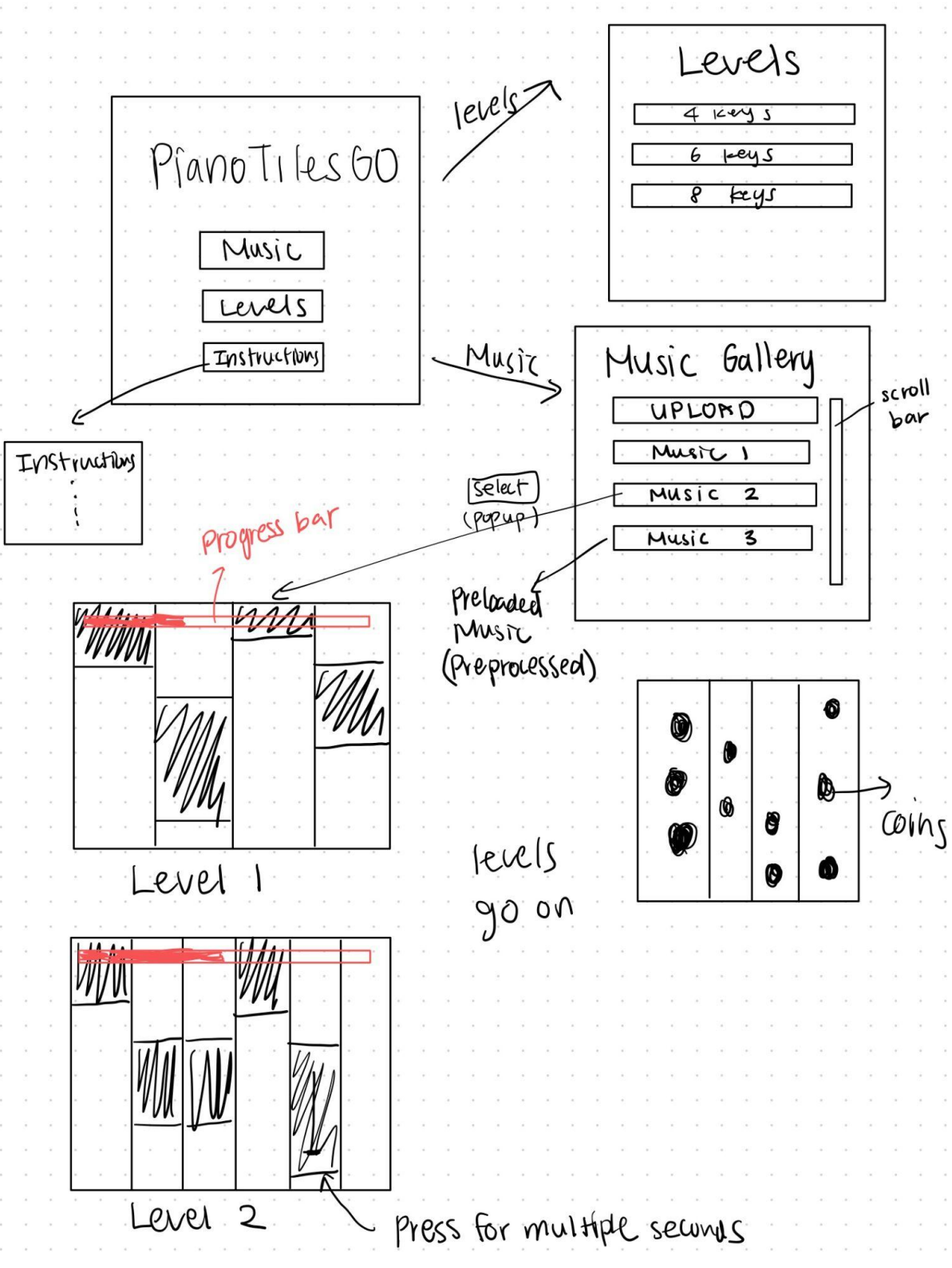
game would allow the user to input custom audio of choice and generate a set of game play that corresponds to the beats, rhythm and notes in the audio. This game trains the player's coordination and practices their hand independence.

- 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.
    - The project that my project is most similar to is the popular piano tiles game. Although our ideas sound similar, I plan to implement levels in not only the difficulty of songs, but also increase the amount of vertical tile bars to be upwards of 8 keys, simulating that of a real piano player. My version of piano tiles would also aim to allow custom audio inputs from the user as opposed to having a static database of a fixed number of possible music selections.
- Structural Plan [2.5 pts]: A structural plan for how the finalized project will be organized in different functions, files and/or classes.
    - Main.py
    - Board.py (board class)
    - Pieces.py (pieces class)
        - Would create multiple instances based on various levels
    - Progressbar.py
    - Levels.py
    - Audio/
        - Sample audio files/data
    - Audio_syncing.py
- 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.
    - The most difficult part of my project is the audio analysis/audio feature extraction portion. This includes tempo tracking, Although an easy option is to use external modules such as Scipy, and librosa, I am aiming to write most of the processing function from scratch since I am interested in learning more and researching signal processing, using spectrogram decomposition and Fourier's Transform. I plan on understanding the algorithms and then implementing the math. I might use numpy to keep track of large audio data in arrays.
    - Another difficult aspect would be keeping track of the time dimension and syncing it to the black piano pieces in real time.
- Timeline Plan [2.5 pts]: A timeline for when you intend to complete the major features of the project.
    - By TP1: Minimal, minimal viable product: Piano tiles game w/o audio. Multi-level (not just four tiles, but more tiles!) randomly generated tiles activated with keypresses. Progress bar. Score tracker.
    - By TP2: MVP: Real time sound and game playing experience. Hardcoded/limited audio choice.
    - Custom audio input. Involves audio processing, feature extraction of pitches and beats.

- Version Control Plan [1.5 pts]: A short description and image demonstrating how you are using version control to back up your code. Notes:
    - I will be using Github to version control my code. I would create a dedicated remote repository to commit, push, and pull from that is connected with my local repository. Below is an example from Hack112.



- Module List [1 pts]:
    - After TP2: Audio analysis such as scipy, librosa, numpy perhaps
- Storyboard


Piano Tiles GO
Music
Levels
Instructions
levels
Levels
4 keys
6 keys
8 keys
Music
Music Gallery
UPLOAD
Music 1
Music 2
Music 3
scroll bar
Instructions
select
(popup)
Progress bar
Preloaded Music (Preprocessed)
Level 1
levels go on
coins
Level 2
press for multiple seconds

1. By TP1: Minimal, minimal viable product: Piano tiles game w/o audio. Multi-level (not just four tiles, but more tiles!) randomly generated tiles activated with keypresses. Progress bar. Score tracker.
2. By TP2: MVP: Real time sound and game playing experience. Hardcoded/limited audio choice.
3. Custom audio input. Involves audio processing, feature extraction of pitches and beats.

https://towardsdatascience.com/get-to-know-audio-feature-extraction-in-python-a499fdaefe42
https://librosa.org/doc/latest/decompose.html