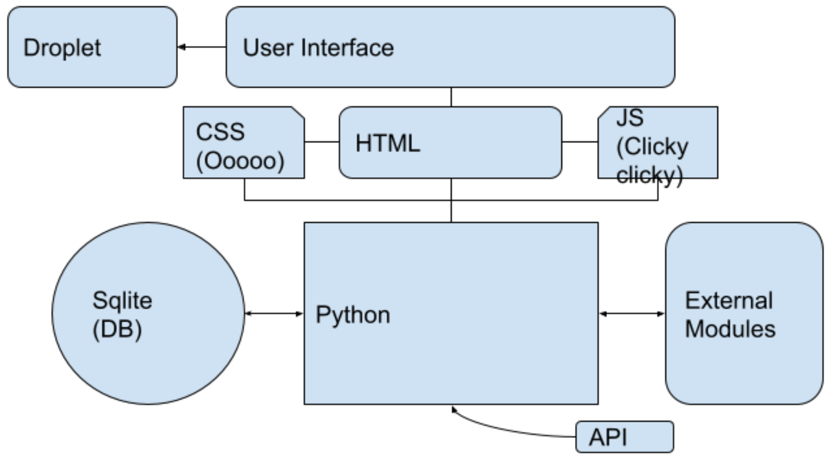
Doink! – Julia (Lia) Nelson (PM), Lucas (LTW) Tom-Wong, Liesel Wong, Tomas Acuna, Dahlia, Tobias, King Hagrid
SoftDev P02
2022-03-04
Title Project: DOINK!

Description of Project:

Rhythm game where the user clicks the screen on pre-ordained intervals. These are randomly generated based on the tempo and time signature.
User Interaction
User clicks anywhere on the screen to register an action
→ If User fails to click on the screen when action is needed or clicks at the wrong time, a life is lost. The will be a delay before ability to lose another life
→ Score increases with each successful click

Components:



CSS: Makes everything pretty(ier).
JS: Interaction and animation.
HTML: What the user sees which will be influenced by everything.
Python: Generate random rhythms, interact with database and external modules, facilitates communication between other components.
DB: Base of data.
External Modules: Maybe? For password hashing or audio interpretation.
Front-End Framework: *Bootstrap*. Framework templates will allow easier formatting over creating custom stylesheets by using something that already exists. The documentation is also easier to understand than Foundation. We will largely be using its preexisting classes, especially with our buttons, in order to provide a consistent look.

What is new compared to P01? **JavaScript!!!!!**
JavaScript helps facilitate user interaction by adding scripts to html allowing the page to react to user inputs.

What is needed for this project? **Audio!**
We need audio for a rhythm game…. Probably. We need to be able to generate a random rhythm as well as audio to compliment it. This will most likely be done through javascript or python.

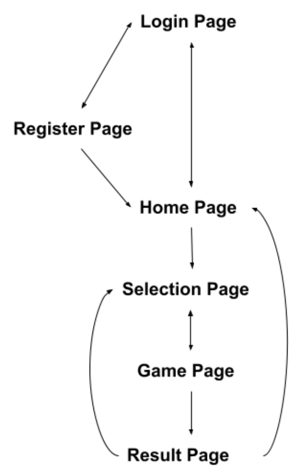
Rhythms
Rhythms will be generated randomly (e.g. eight notes, triples, whole notes)
→ They will be generated in rounds
    → A round will most likely contain 8 bars
→ Background sound will be generated to match the rhythms
→ Generated indefinitely till the user ends or loses

Canvas
→ Visuals will consist of a large canvas that the user can click on
→ Will react to clicks
    → Clicks will register in order to consider a note being played
        → This will also lead to a sound being played

Database:

User (Primary Key, Text)	Pass (Text, Ideally Hashed)	High Score (Int)

Site Map:



Understanding the Site Map:

- Register Page
  - Button to Login Page
  - Input
    - User
    - Pass
    - Confirm Pass
  - Button to enter input
    - Gives error or leads to home with user logged in
- Login Page
  - Hyperlink to Register Page
  - Input
    - User
    - Pass
  - Button to enter input
    - Gives error or leads to home with user logged in
- Home Page
  - Logout button
  - Welcome user
  - Instructions for game
  - Button to start game (leads to selection page)
  - Leaderboard

- <u>Selection Page</u>
    - Select tempo
    - Select time signature
    - View current score
    - Button to Result Page (End Game)
    - Button back to home page
    - Button to Game Page
- <u>Game Page</u>
    - Display rhythms
    - Cursor to follow rhythms and get user input
    - Interact with user input
    - Display tempo
    - Display time signature
    - Display lives (3 lives, if you make a mistake you lose a life)
    - Display score on the side
    - When the game ends (Loss due to lives or if ended game), display score
    - Button to Result Page (end game)
- <u>Result Page</u>
    - States if high score
        - Asks to save score in database
    - Button to Home Page (home)
    - Button to Selection Page (start new game)

Goals:

1. Get a metronome playing
    a. Make metronome optional
2. Show tempo and time signature
3. Generate rhythms
    a. Have maximum length of time
    b. Determine time left
    c. Evaluate feasible rhythms
    d. Select rhythm from list
    e. Stop at end of time
4. Display rhythms generated (bars and notes should be shown)
5. Add cursor
6. Determine user responses and evaluate if correct
7. Display result of said evaluation
8. Add other tempos

9. Add other time signatures
10. Establish point scoring system
    a. Three lives
    b. Given a grace period before they can lose a life again
    c. Each correct rhythm gets a points
11. Display lives and points on screen
12. Make accounts
13. Save past high scores
14. Hash passwords
15. Generate background sound to match rhythms

Make optional

Roles:

Lia (PM): Audio/Rhythm Generation (Create Functions for JavaScript to Call)
LTW: HTML + CSS (Design, Front-End)
Liesel: Sqlite and Python Interaction (Account-Based Interaction)
Tomas: JavaScript (User Interaction and Animation)

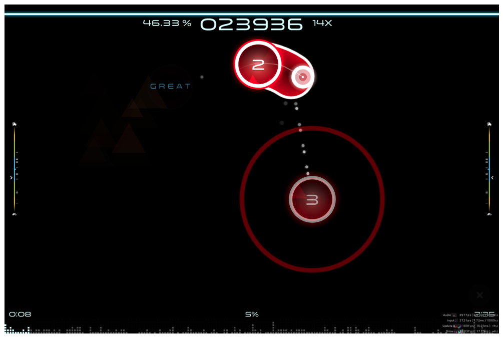
APIs:

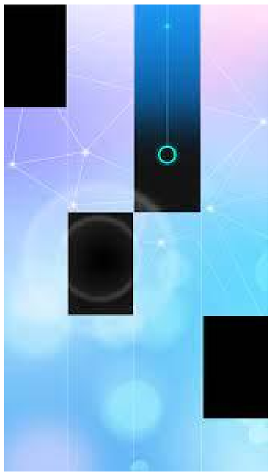
- Web Audio API
    - Allows for more complex audio processing as well as playback
    - Will be utilized to create a metronome
- GetSongBPM API
    - Allows for searching for songs in a particular tempo and time signature
    - Will allow for us to supplement the user with an option to have a backing track (in addition to or instead of the metronome) when available

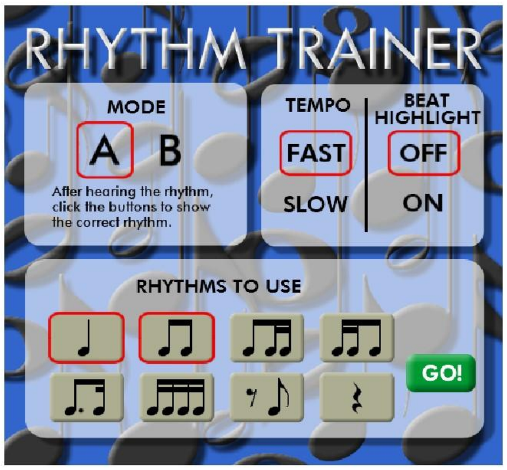
Potentially Important and Useful Links:

https://developer.mozilla.org/en-US/docs/Web/API/HTMLAudioElement/Audio
https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API
https://developer.mozilla.org/en-US/docs/Web/API/HTMLAudioElement
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio
https://en.wikipedia.org/wiki/WAV
https://docs.python.org/3/library/wave.html
https://piazza.com/class/kv0wqn7faux3ye?cid=33
https://developer.chrome.com/blog/canvas2d/
https://github.com/tabs4acoustic/GetSong-Metronome
https://getsongbpm.com/api?fbclid=IwAR1PKDUzz218Cm0QLaC53IIoOTAhjgQ2Wp284T5TL16IB0CoE9Af7FdSUQ4

Inspiration:







MODE [✓ A] B

After hearing the rhythm, click the buttons to show the correct rhythm.

TEMPO [✓ FAST] SLOW

BEAT HIGHLIGHT [✓ OFF] ON

Target Ship Date:
2022-03-28