Names: Jake Colleran, Thomas Ferrell

December 7, 2022

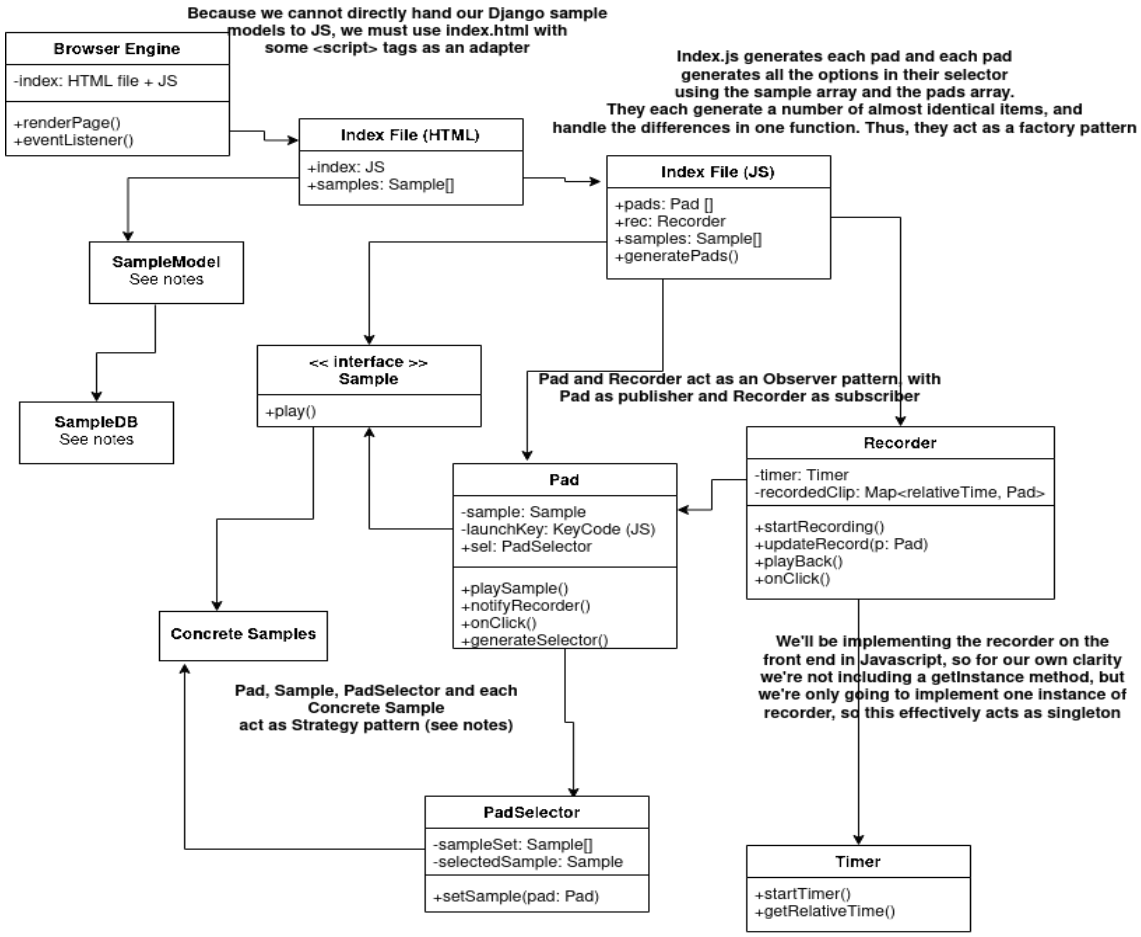
Final Project Report

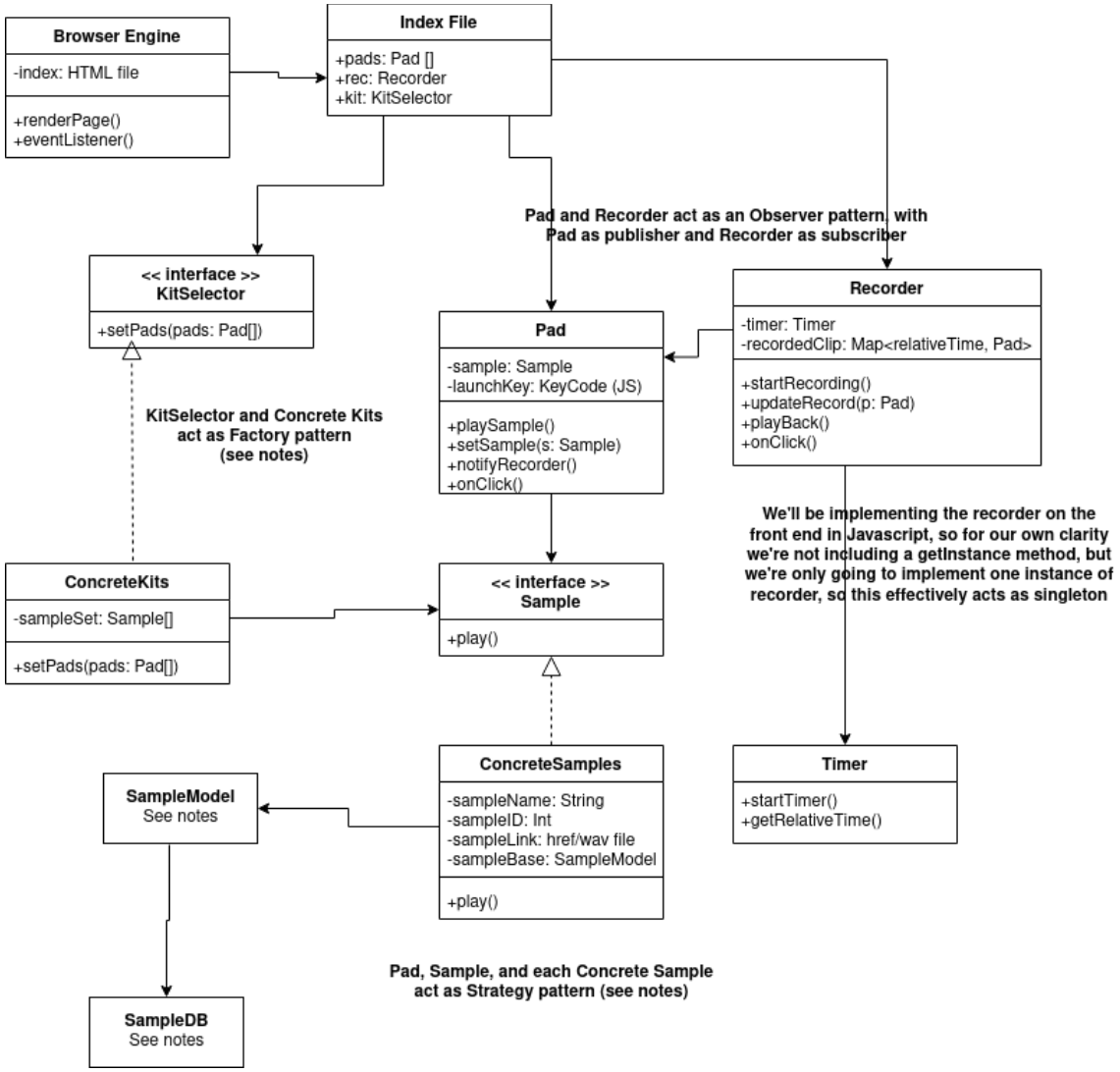
1. Name of project: Tap Time, Names of team members is in the header.

2. The features that have been implemented so far include the recording button, which our team said was only going to be able to record one sequence of beats at a time. If the user was to press the clear button, then the user would not be able to go back and edit the track. Our team also implemented individual drum pads for the sound samples. Originally the plan was to have six in project five, but we ended up having nine to try and accommodate the list of sound samples and make a bigger "drumset". Each drum pad has a unique effect that when pressed, will light up. This happens when the user presses the drum pad and when the user presses the play button after hitting the record button. Our team was also able to implement a Django backend manipulated by Python so that we could access a wider range of sound samples. In order to get the sound samples into our index file, it was necessary to take the sound samples and turn them into an array of all the different sound files, so that our Javascript file could actually access the sound samples. Django is better with "static" pages instead of data that needs to be dynamically loaded (This is mentioned in the class diagram down below). If the data does not necessarily need to be changed or in our application, swapped out, then it could be a great fit. Unfortunately, this led to our team deciding not to implement the drum kit selector as the way the samples are actually able to be passed from the Django database to Javascript is rather tedious. Along with timing issues from Javascript, this could definitely cause some headaches for the project, so we decided to not include it.

The Final Class Diagram (First) vs. Project Five Class Diagram (Second)

3.


Because we cannot directly hand our Django sample models to JS, we must use index.html with some <script> tags as an adapter
Browser Engine
-index: HTML file + JS
+renderPage()
+eventListener()
Index File (HTML)
+index: JS
+samples: Sample[]
Index.js generates each pad and each pad generates all the options in their selector using the sample array and the pads array. They each generate a number of almost identical items, and handle the differences in one function. Thus, they act as a factory pattern
Index File (JS)
+pads: Pad []
+rec: Recorder
+samples: Sample[]
+generatePads()
SampleModel
See notes
SampleDB
See notes
<< interface >>
Sample
+play()
Pad and Recorder act as an Observer pattern, with Pad as publisher and Recorder as subscriber
Recorder
-timer: Timer
-recordedClip: Map<relativeTime, Pad>
+startRecording()
+updateRecord(p: Pad)
+playBack()
+onClick()
Pad
-sample: Sample
-launchKey: KeyCode (JS)
+sel: PadSelector
+playSample()
+notifyRecorder()
+onClick()
+generateSelector()
Concrete Samples
Pad, Sample, PadSelector and each Concrete Sample act as Strategy pattern (see notes)
We'll be implementing the recorder on the front end in Javascript, so for our own clarity we're not including a getInstance method, but we're only going to implement one instance of recorder, so this effectively acts as singleton
PadSelector
-sampleSet: Sample[]
-selectedSample: Sample
+setSample(pad: Pad)
Timer
+startTimer()
+getRelativeTime()



Notes: We ended up bailing on the kit selector, as we faced issues with Javascript timing that took a long, long time to debug. The code is a little bit bloated with unnecessary patterns - we both decided to try to work with a new tech stack, and discovered Django is much more suited to websites with “static” pages - i.e. they can be dynamically generated, but they are non-interactive. We ended up having to port over the Django model in a hacky way to the index.js file, as Django doesn’t have any good mechanisms for templating in an external js file. Any other solution we tried would’ve required a page reload on every change. If we were to do this again with what we know now, we would likely just use AJAX and a well-organized webserver; Django would be better for a site such as sharing samples, but not a drum machine like this.

The other major issue is that Javascript is challenging to track and debug, at least for timing and event-related things. There were 2-3 am nights spent debugging, with the console giving no valuable information as values would seemingly change with no reason to, and we would ultimately find we missed one state variable mention. I’m sure

javascript has some better mechanisms for tracking this stuff, but I'd definitely need some more time to learn the language a bit better before trying to do something time-sensitive again.

Key changes for this system include that we have removed the drum kit selector and only changed the code for the drum pads and index file so that they could correctly interact with the Django database sound samples. From project five to project six, the UML for the index file, browser engine, drum pad, recorder, and timer, stayed the same.

4. Third-party code for this project and what is originally included:
    a. https://www.youtube.com/watch?v=HTTaO1IjjlM This is a video about setting up a simple drum pad in a browser using a live server extension, React, Javascript, HTML, and CSS. Our team used this code to set up the drum pads and then added on by adding the recording feature, and more buttons that can be set to specific sound samples, so that they can be dynamically swapped out.
    b. https://www.bootstrapcdn.com/ , https://babeljs.io/docs/en/babel-standalone Other outsides sources that were used more for styling and compatibility across different browsers were Bootstrap CDN (Styling, taught in CSCI 3308), ReactJS, Babel standalone, which converts modern es6 Javascript into say es5 Javascript if needed for that specific browser (It is recommended that you use babel to transpile the Javascript code ahead of time instead of having to transpile every time when the webpage is requested). The reason this happens is that es6 Javascript is not compatible with every browser, so this allows us to write the code without worrying about making sure it will be compatible with every browser platform. ReactJS allows us to use patterns like the state pattern in a sense. ReactJS allows us to give Javascript the ability to essentially keep track of the state of the webpage when using React.useState() function. ReactJS is what is used to let the drum pad know when to flash the color of the drum pad, when the application is recording, etc.
    c. The stuff that is original for this project is simple styling changes, Django database integration to allow sound samples to be swapped out dynamically, adding timing to the recording feature so that it would know exactly how long the last beat was played until the next beat is supposed to be played, and the sound sample selector that uses a Javascript array passed in from HTML that got the sound samples from the Django database to then set a specific sound for the specific drum pad. Each individual drum pad now has the unique feature to play backlighting when playing back the recording. In the example from YouTube (Part A), the light for the drum pad was implemented, but not for playback and not for correct timing.
5. Statements on the OOAD process for Tap Time
    a. One key design process element that our team has experienced with this Tap Time project in terms of object-oriented design is that a lot of the frameworks we use already try to set up the code for the developer so that it is already in an object-oriented fashion. That being said, our team was able to use these already implemented features across our whole project to make it implement these design patterns in a unique way.

b. One key design process element that our team has experienced that was beneficial towards our team actually getting the project done was the ability to create a very detailed UML class diagram and then stick to that UML class diagram as much as possible as the project experienced changes when our team realized that Django might work better for “static” pages that do not need data dynamically loaded at runtime and that timing for Javascript is a bit complicated so extending off of the drum pads capability becomes a bit hard when the timing of the state is not universal. Overall, Javascript did serve as a very good functional language that uniquely gives us class capabilities with methods and variables. The UML class diagram changed a bit, but only because of our preconceived notions of the Django database and Javascript capabilities. Our team definitely has a more solid understanding of the Django framework and Javascript
c. Another key design process element that our team has experienced that was not the best for our team’s development was running into weird dependency issues for Javascript. Our team used a React useEffect hook to load in event listeners for pressing keys on the keyboard and connecting those to the functionality for clicking different pads. A typo in the initial useEffect ended up loading a pre-compiled version of a function at the first render, but this pre-compiled function had a dependency on a flag defined in a higher scope. This resulted in a bug where recordings continued to record even when a recording was stopped. This definitely held back our team for a bit in finding out exactly why React useEffect was not setting the recording state back to false when the recording button was pressed by the user.

Overall this project has been a blast for our team and if our team was to ever do something like this again, we would of course restructure the architecture of the application with the knowledge we have learned over this project in mind, especially with Django database servers and Javascript. If we were to do a web-based application, our team would still probably stick with Javascript but swap out the Django database for an organized web server and AJAX calls, or our team might do an actual application/software instead of a web-based application so that we could focus more on the functionality of the program rather than dealing with the connections between Javascript, the Django database, and React components. Creating a software/application would allow us to possibly use different programming languages like C++ to implement those patterns in a different way. This project definitely showed our team how many frameworks actually try to set it up in an object-oriented programming way.