VioletMIDI: A portable, user-friendly app for writing music.

Capstone Project Legacy & Specifications (Humza Khan)

Updated 5 December 2023 (Final Version)

1. Project Description & Purpose

As a musician, I often find myself coming up with ideas for songs while on the go, unable to properly record them. I'd like to create an app that fulfills this purpose: a simple DAW/MIDI editor that can run on both mobile and desktop devices. This will be a semester project as I intend to graduate this winter.

2. Initial Expectations

- I will develop this app primarily with mobile devices in mind (see schematic below) but it should work on desktop too. To build a cross-platform mobile app I will use the following technologies:
    - Front-end: HTML, CSS, and JavaScript (Canvas API)
    - Program logic: JavaScript (Tone.js API)
- Implementation of basic MIDI editor features such as:
    - multiple tracks
    - basic instrument selection
    - manipulation of individual notes to create melodies and harmonies
    - saving and loading of project files
    - playback (ability to listen to the current project file)
- Easy-to-use, accessible UI for users of all musical/technological skill levels:
    - visually appealing interface
    - gestures for mobile users
    - desktop users can just use point-and-click or keyboard shortcuts

3. Current State of the Project

- Basic MIDI functionality (note manipulation)
    - the user can now freely scroll/zoom on both axes (this actually took several hours)
    - select individual or groups of notes
    - place, move, resize, and delete notes
    - copy and paste notes
- Multiple tracks
- Basic playback
    - can choose from a few different preset synthesizers
    - optimized playback code so the program no longer randomly lags out during playback
    - can play many notes at once (chords, etc.)
    - can play the project a single time, or select a region to play on repeat (useful for songwriting)
- Project management (change tempo, save, etc.)
- Tested on 2020 M1 Mac and Samsung Galaxy S21 (see Fig. 2) — works mostly fine!
    - Gestures are still missing and the layout is still a little clunky for mobile phones. At this point I will probably just abandon the mobile idea. (The app will still be cross-platform, it will just work better on a computer.)

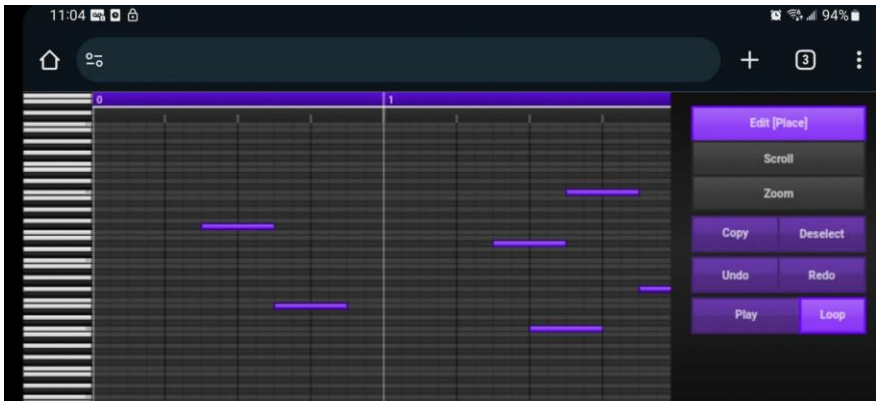

Figure 2: *How the app looks on my smartphone.*

4. Remaining Areas of Concern

- One known bug: sometimes after changing project settings (song tempo or length), playback fails. This can be remedied by saving the project, refreshing the page, and reloading the project.
    - This seemed to be an internal Tone.JS bug and I couldn't figure out how to fix it. I believe it has to do with which version of Tone.JS I'm using.
- Future directions: This is a completed one-semester project and for the purposes of this course, I've finished most of the things I set out to do. However, I would like to continue working on this app, and I have a few ideas for where it could go next:
    - **More in-depth instrument selection:** While the program does allow you to create multiple tracks and assign to them different instruments, these are limited to various synthesized waveforms. To add a library containing real instruments like the piano and guitar, I would need to acquire, process, and organize hundreds of samples of real instruments — a process that would take a great deal of time, but would surely add richness and depth to the product.
    - **Regions, an intermediate unit between notes and tracks:** Many modern MIDI sequencers organize individual tracks into "regions" of MIDI notes, which can be moved around, copy/pasted, and looped. Although this is an idea I had wanted to include from the start, implementing it simply took too long.
    - **Undo/Redo history:** This would be quite a useful quality-of-life feature to have for an app like this. The easiest way to do this would probably be to store all of the user's operations in a list, representing each operation $F(x)$ on the current track x as a collection of the characteristic parameters of that operation, such that the program can easily determine the inverse of that operation $F^{-1}(x)$ where $F^{-1}(F(x)) = x$. For instance, to move a note from position 4 to 6, store the operation $\{\text{type: "move", from: } 4, \text{ to: } 6\}$, and to undo, execute the inverse of that stored operation, which would be $\{\text{type: "move", from: } 6, \text{ to: } 4\}$.
    - **More efficient data serialization:** Right now, files are saved and loaded as text files. Song data (which consists of the basic project information alongside a list of each track and its associated list of notes) is converted to a string format and saved to a text file. When the user loads this file again, the program decodes the text to reconstruct the song. While this generally works well, it's not a particularly elegant solution, nor is it appropriate/scalable for large songs with several tracks and hundreds of measures. In the future I would like to create an algorithm that instead converts and saves the song data as some sort of binary data format, which would be much more efficient. I have a general idea of how I would do this, but it would definitely take a while to implement and test thoroughly.

5. Activities and Time Logs

Task	Time
Project proposal & legacy	3 hours
Initial research on APIs	3 hours
App design planning / drawing schematics	2 hours
UI design and style	5 hours
UI programming / rendering	15 hours
Tone.js API integration	4 hour
Playback programming	7 hours
Code cleanup/documentation	9 hours
Graphics	3 hours
URC Abstract & Poster	4 hours
Total	**55 hours**

6. Technical Lessons Learned

Cross-Platform Development

Developing an app that feels responsive on both desktop and mobile devices has been a particularly difficult and time-consuming task, presenting challenges such as adaptive UI layouts and gesture programming. Since this is only a semester long project, I may focus only on the desktop app to ensure I can deliver as complete a product as possible.

UI Programming & Libraries

In terms of hours spent programming, the bulk of my time has been spent on crafting a responsive, bug-free UI that looks good. I suspect this may be due to my decision to build the project from scratch, with minimal assistance from external libraries or APIs. For future projects, perhaps a UI library would save some time.

7. Managerial Lessons Learned

• I strongly prefer to work by myself, but this has been a very ambitious project to do alone. Perhaps working with 1 or 2 more group members would have made the work easier. Even just the utility of having extra devices to test the software on would have been helpful.

8. Recommendations to Future Projects

• It's OK to use external libraries/APIs if it'll save a lot of time and headache — you don't have to reinvent the wheel every time you code something.

• Pick a topic or project idea that interests you. While this project was certainly a challenge to complete, I had a lot of fun with it, and it has been very rewarding for me to see how my finished product develops. As a lifelong musician pursuing a career in computer science, music technology has always interested me — if I had picked a project I was ambivalent about, or one I didn't particularly care for, I'd imagine completing this project would have been much more difficult.