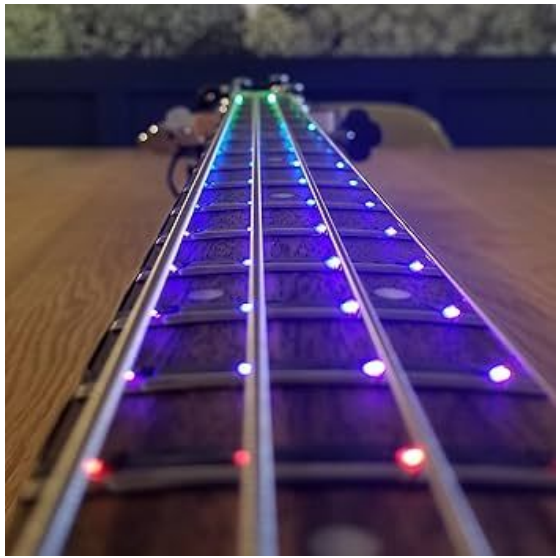
Carnegie Mellon University

Team A2: SuperFret

Owen Ball, Ashwin Godura, Tushaar Jain

Use Case

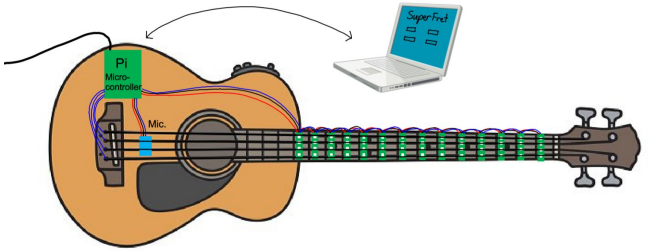
When trying to learn guitar, beginners face challenges trying to learn how to finger notes and maintain rhythm



The SuperFret system shall:

1. ***Visually indicate notes***
2. ***Detect finger positions***
3. ***Facilitate learning scales***
4. ***Provide feedback***
5. ***Be intuitive to use***

Design Requirements



Hardware

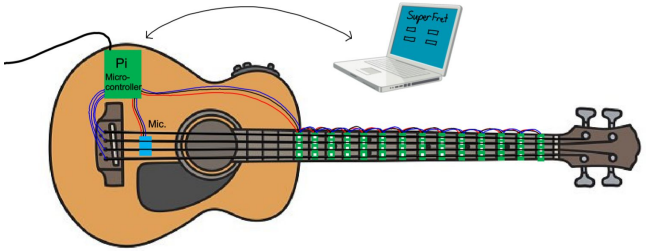
- ≥ 56 individually addressable LEDs (14 frets, 4 strings).
  - ***"Visually indicate notes"***
- Can light up ⅔ of LEDs at half brightness
  - ***"Facilitate learning scales"***
- Audibly (~60bB) indicate target tempo
  - ***"Be intuitive to use"***
- < 1 mA through body max (based on IEC 60479-1)

Firmware

- <50ms latency from strum to LED response
  - ***"Visually indicate notes & Provide feedback"***
- Support 100 beats per minute and down to 1/8th notes

Carnegie Mellon University

Design Requirements



Web Application:

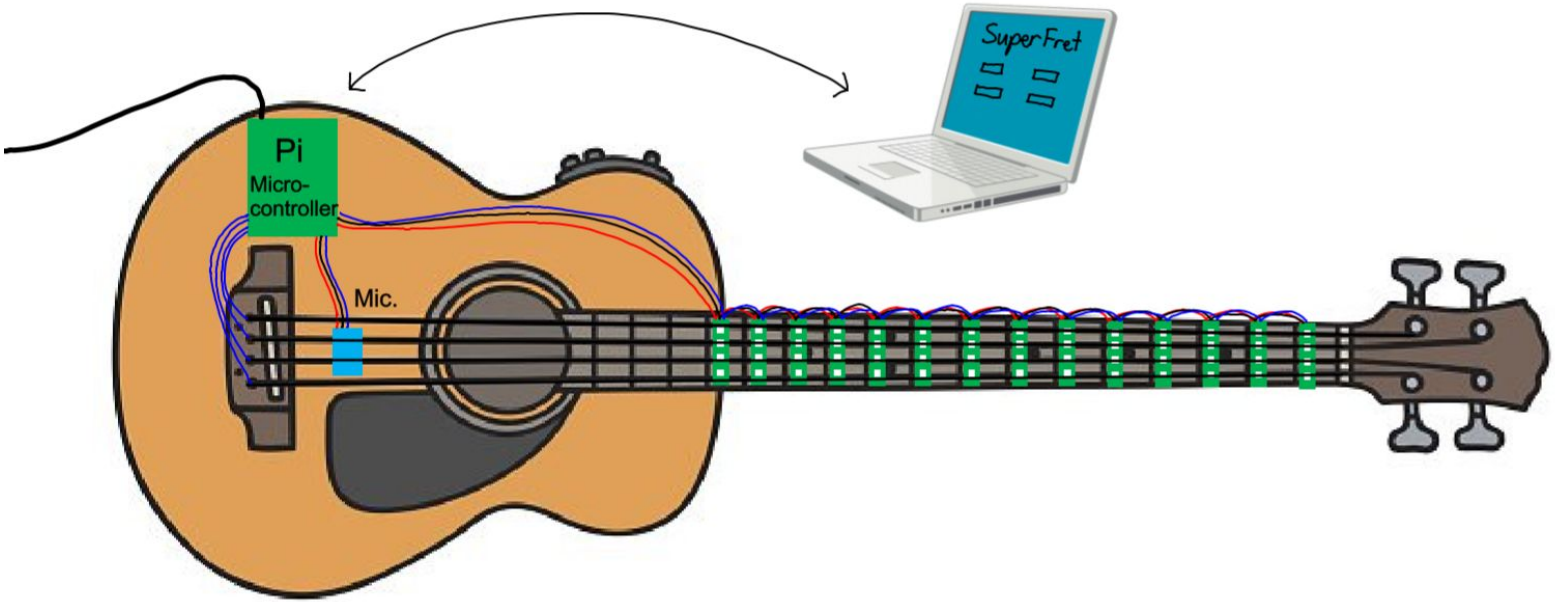
- Start and stop routines on the guitar
- Add up to 1GB of user's own MIDI files
- Display practice statistics (rhythm score, accuracy score)
  - ***"Provide feedback"***
- < 0.25 second network delay

System Accuracy

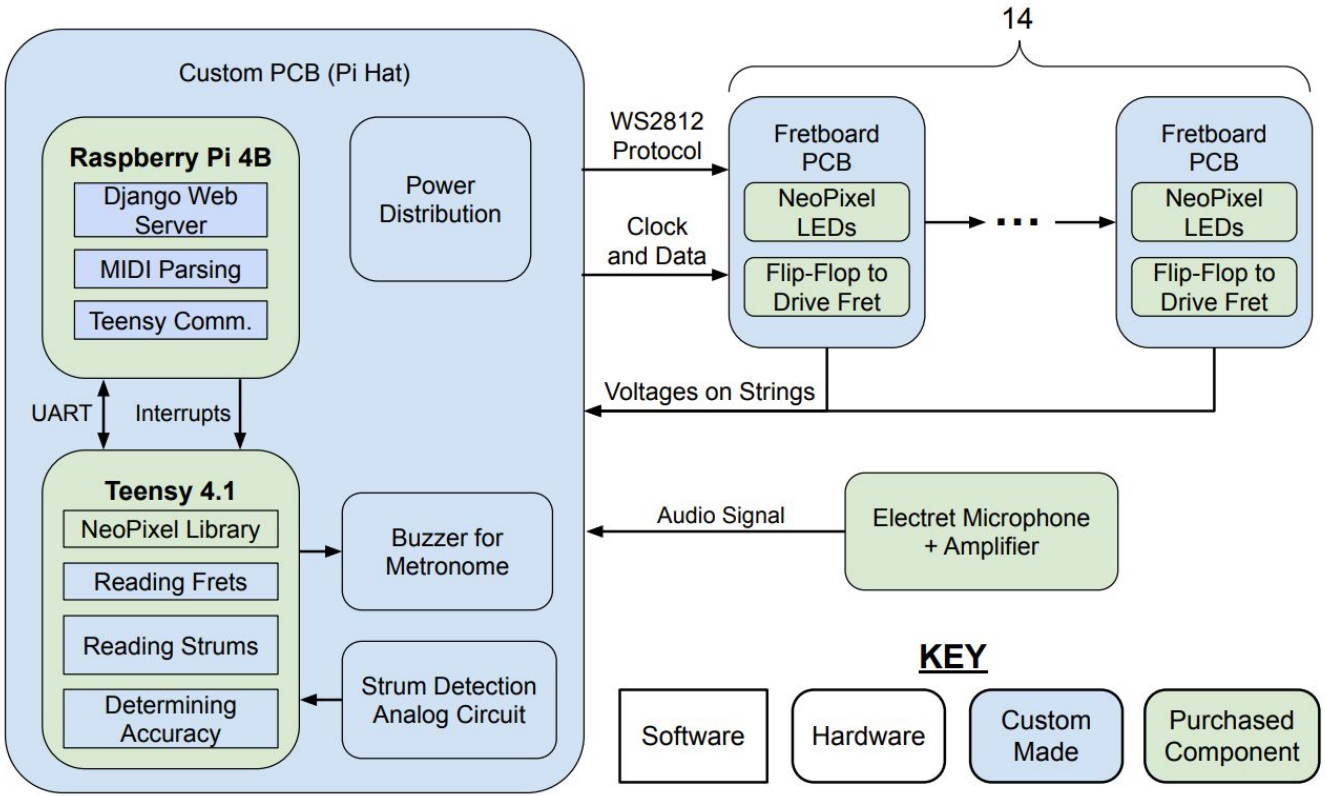
- 99% accuracy for detecting finger placement and strumming
  - ***"Detect finger positions"***
- 100% accuracy for lighting up the correct LED(s) for a note or scale

Carnegie Mellon University

Solution Approach

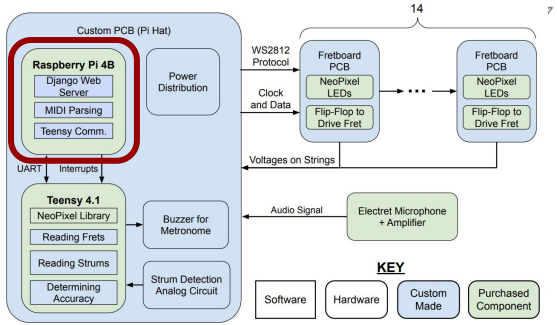


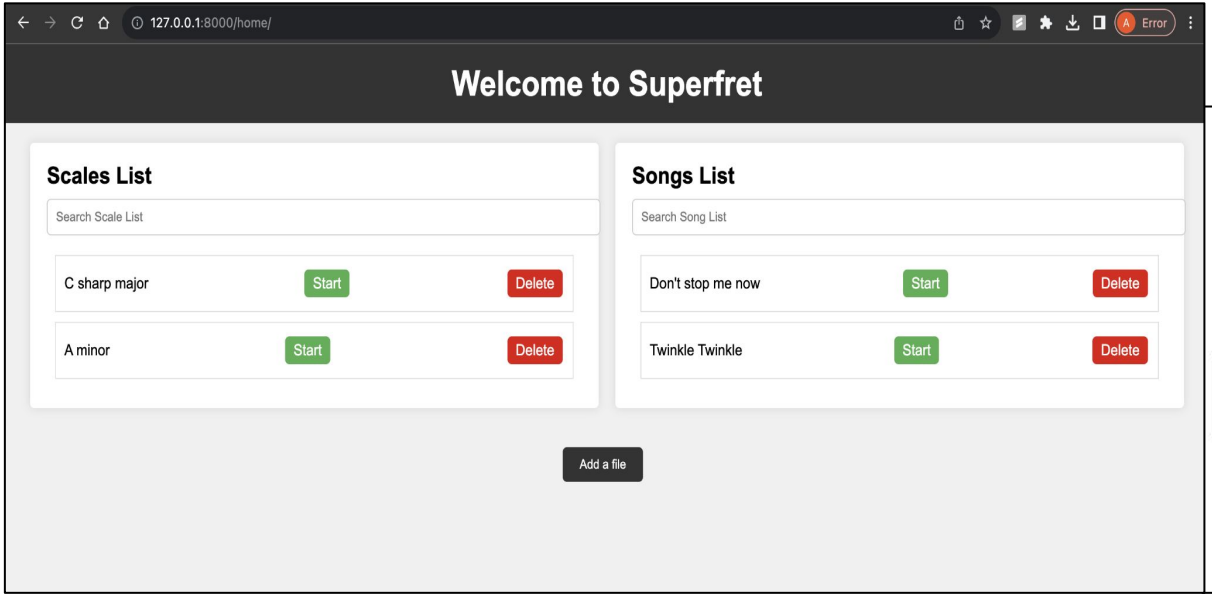
Overall Block Diagram

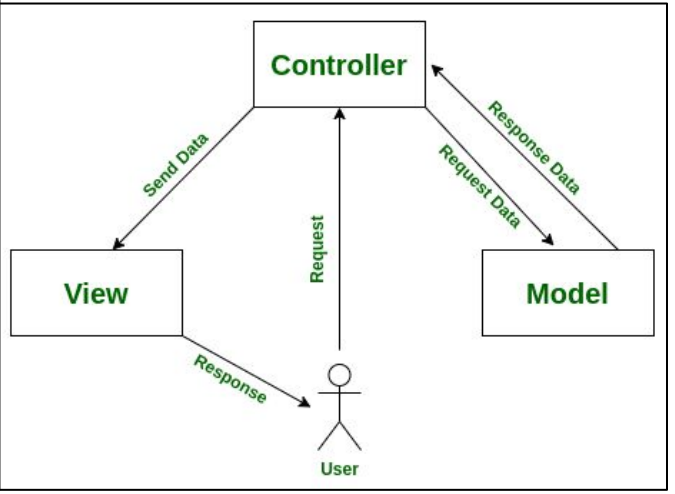


Carnegie Mellon University

Implementation - Web Application





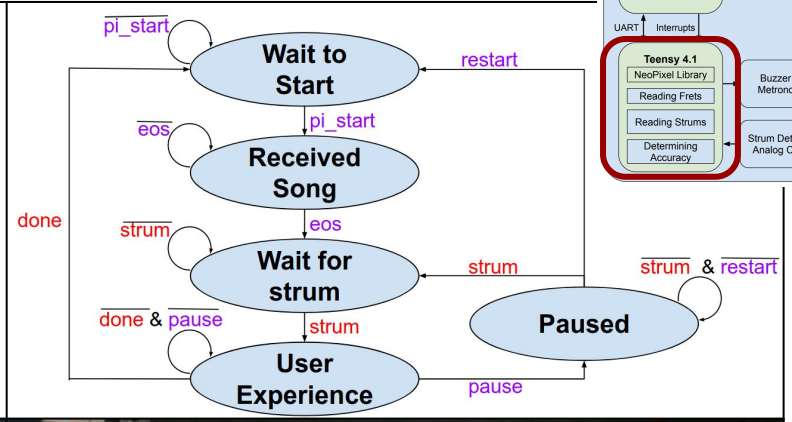


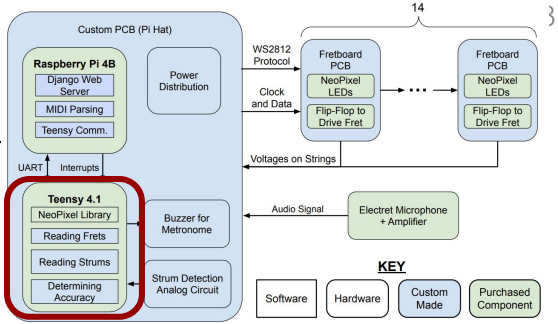
Carnegie Mellon University

Implementation - Microcontroller

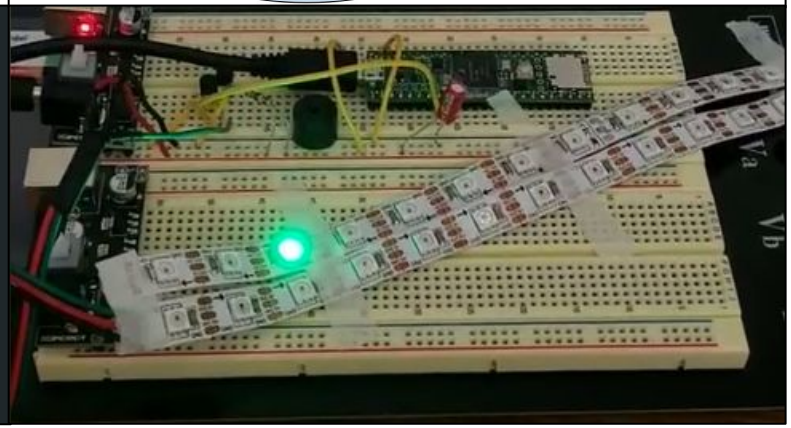
4D 54 68 64 **(MThd)**
00 00 00 06
00 01 (format 1 = one or more simultaneous tracks)
00 03 **(3 tracks)**
01 80 **(0x180 = 384 ticks/quarter note)**

4D 54 72 6B **(MTrk)**
00 00 01 D2 **(Chunk length 0x01D2 = 466 bytes follow)**
00
FF 58 04 04 02 18 08 **(time signature)**
00
FF 51 03 08 52 AE **(tempo = 0x0852AE = 545,454 us/beat ⇒ 2.1812 seconds / measure ⇐⇒ 110 BPM)**
00
FF 03 15 45 6C 65 63 2E 20 50 69 61 6E 6F 20 28 43 6C 61 73 73 69 63 29 **(Elec. Piano (Classic)) - sky blue**
00
C0 00 **(Program Change on ch 0)**
00 **(0 delta MIDI ticks)**
90 3C 32 **(0x90 = Note ON event, 0x3C = Note 60 (C4), 0x32 = 50 velocity)**
81 40 (Variable length encoding **1000|0001 0100|0000 ⇒ 00000011000000 = 192 ticks)**
80 3C 00 **(0x80 = Note OFF event, 0x3C = Note 60 (C4))**



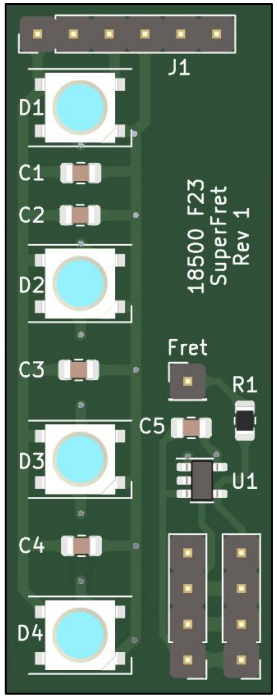


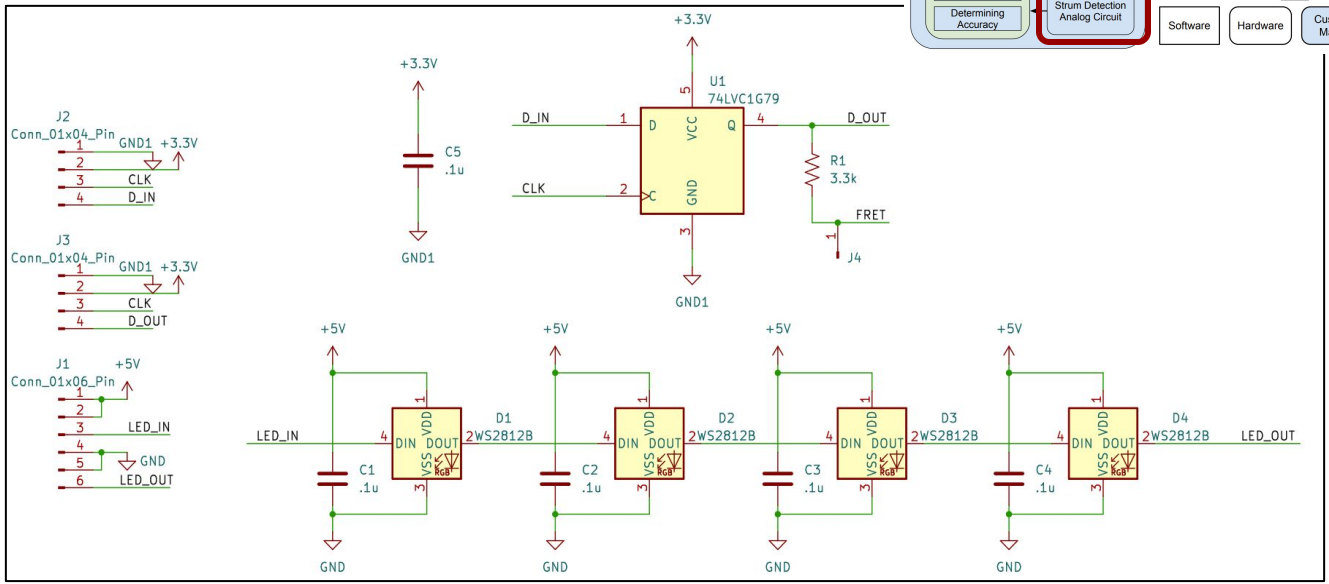

```
void parseMIDI() {
  int NUM_BYTES = sizeof(MIDI);

  bool parsed_MThd = false;
  uint32_t bytes_in_MThd_header = 0;
  bool parsed_bytes_in_MThd_header = false;
  bool parsed_bytes_in_MTrk_header = false;
  bool parsed_ticks_per_quarter_note = false;
  bool parsed_MTrk = false;
  uint32_t bytes_in_MTrk_header = 0;

  NUM_NOTES_FOUND = 0;
  MICROSECONDS_PER_BEAT = 500'000;  // default
  TICKS_PER_QUARTER_NOTE = 48;      // default
```

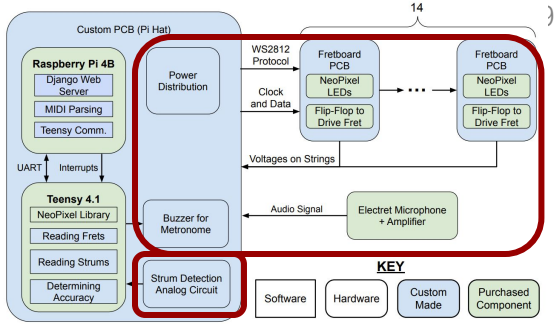


Carnegie Mellon University

Implementation - Hardware







Carnegie Mellon University

Testing, Verification, and Validation

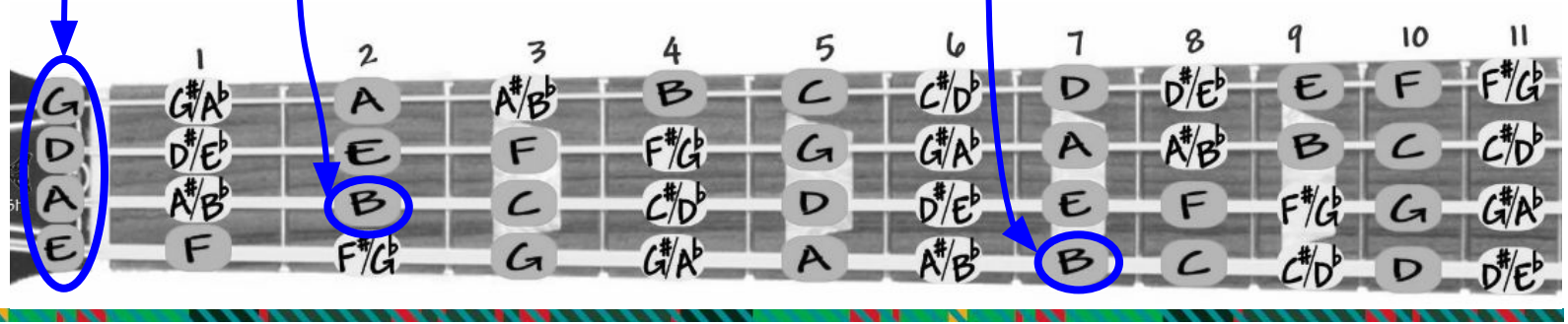
Latency	Hardware: Oscilloscope to measure delay between stimuli, such time from strumming to LEDs being written to Webapp: Measure one-way latency using time stamped requests
Accuracy	Strums: Play 100 1/8th notes at 100 BPM on each string and record the ambient sound level. Finger Placement: Place a finger on each combination of string and fret position and monitor serial port. LEDs: Light up each LED white and verify that the proper LED lights up. Verify current when illuminating all LEDs white at 50% brightness is <1.5A
User Experience	Have users evaluate categories on scales from 1 to 5 to create a quantitative metric Webapp: Intuitive interface, easy to read statistics, intuitive uploading of songs, etc Hardware: Comfortability, effectiveness of LEDs, volume and pitch of metronome

Risks Mitigated

- Lack of web app experience. Switching from Flask to Django
- Understanding MIDI files

New Challenges

- Detecting open strings
  - There is no fret-string contact
- Multiple ways to play the same note
  - How to choose where to instruct the user to play?



Carnegie Mellon University

Project Management