‘Sing us a song, you’re the piano pi’

Talking Piano

Team B2
Angela Chang, John Martins, Marco Acea

—

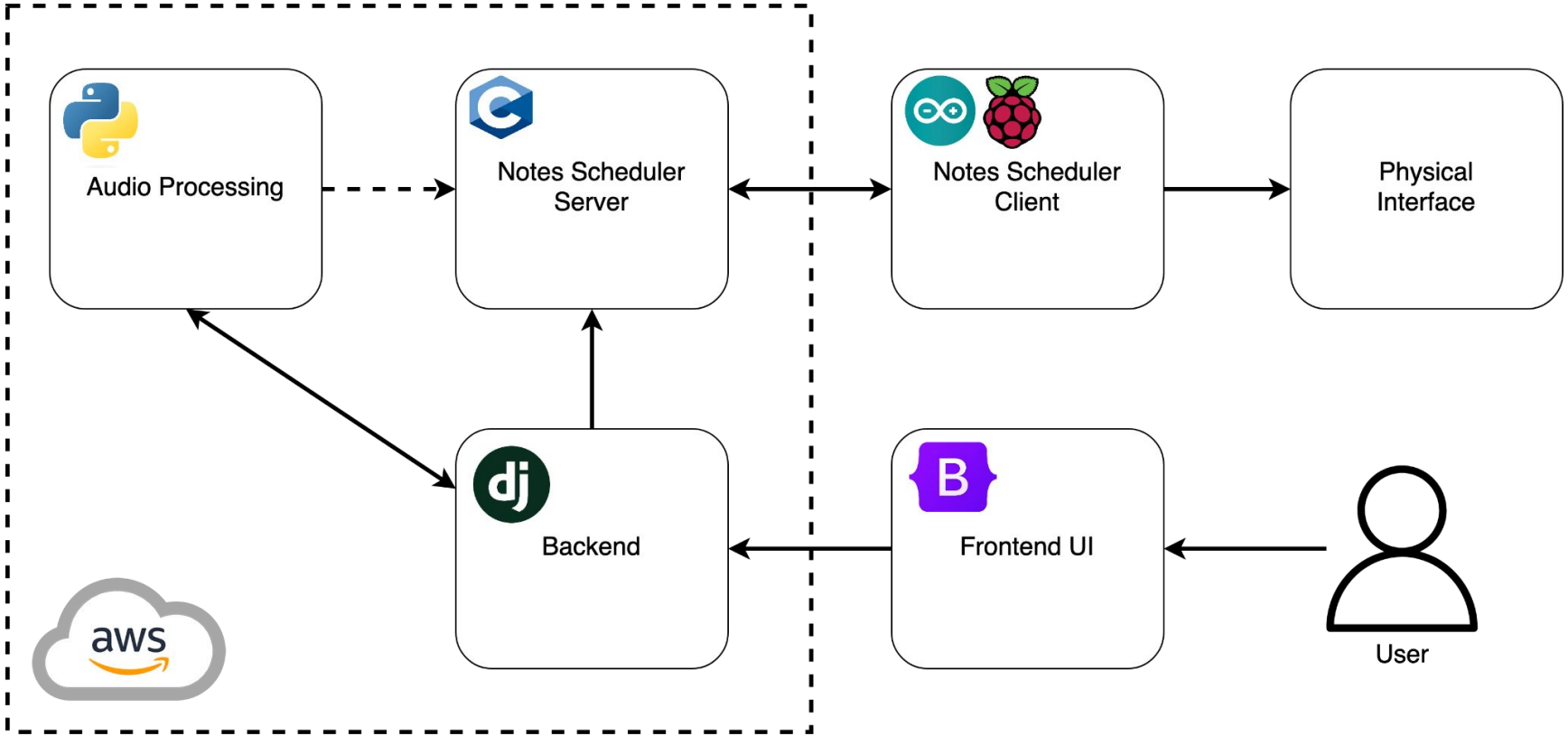
Use-Case and Requirements

Explore music beyond physical constraints by creating human speech on a piano!

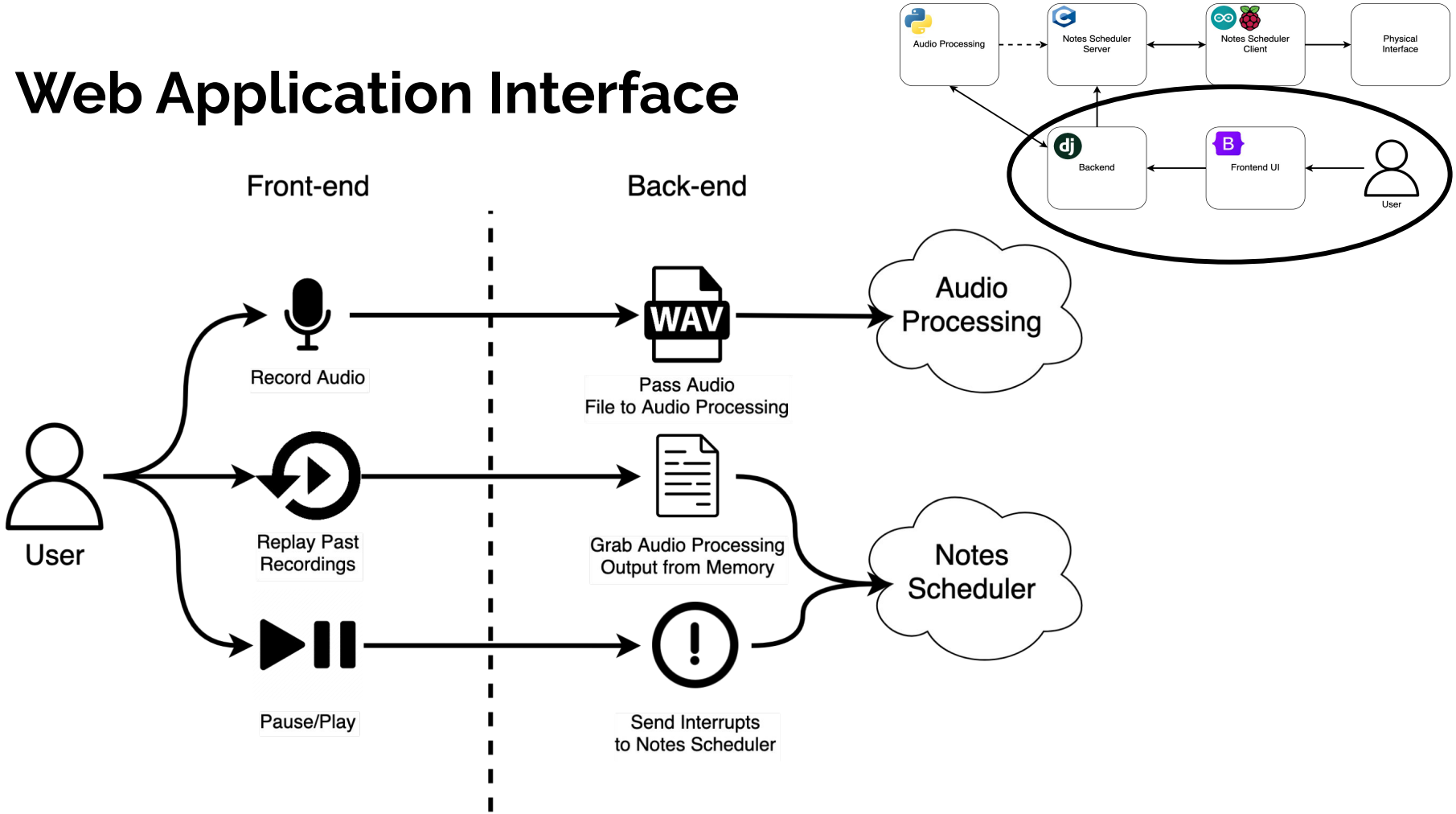
- Record user speech (via UI, that also offers playback features)
  - 200ms end-to-end latency
- Convert input speech into piano notes
  - 80% Frequency extraction accuracy
- Schedule those notes onto a piano
  - <5% of syllables missed (delayed/elongated/sped, not dropped)
- Implement a physical device that can press the keys on a piano
  - 80% Fidelity Rate

ECE Areas: Software Systems, Signals and Signals

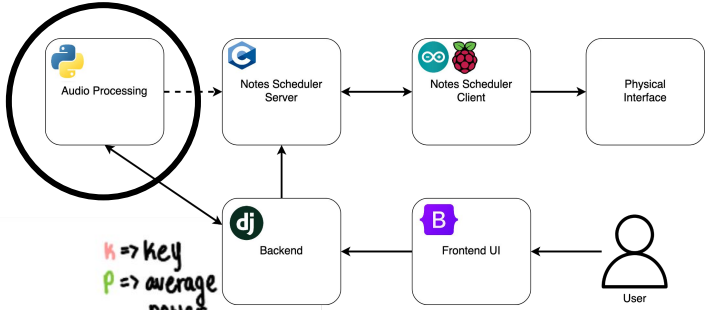
Solution Approach



Web Application Interface

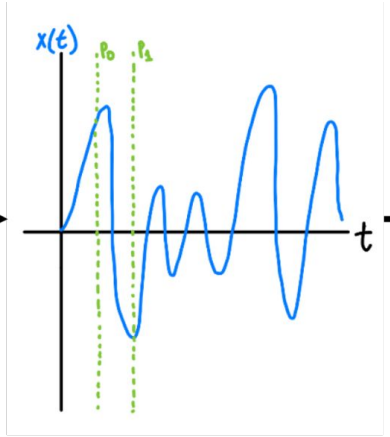


Audio Processing

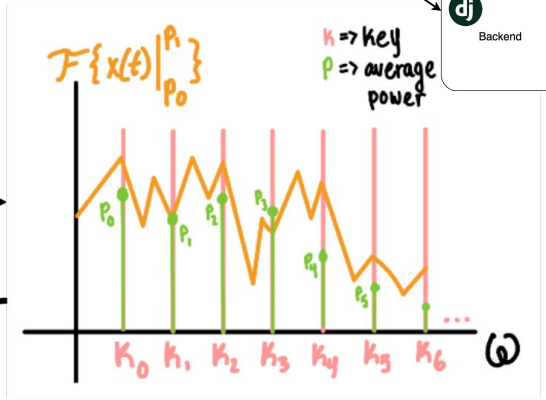


UI
Backend

Incoming Audio Recording



Extract Time Series Data from .wav file



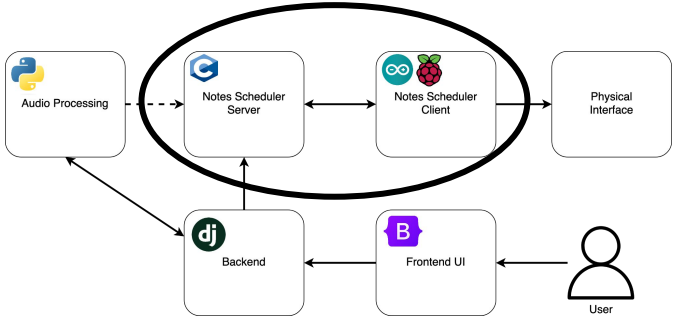
Perform Fourier Transform to extract key frequencies

```
Time | (Key, Power)
0    | (0,70) (1,65) (2,70) ...
1    | (3,25) (4,30) ...
2    | ...
3    | ...
10   | ...
...
```

Generate text file with information about what keys should be played at what time

Notes Scheduler

Notes Scheduler



Server-Side

Client-Side

Audio Processing

Time	(Key, Power)
0	(0,70) (1,65) (2,70) ...
1	(3,25) (4,30) ...
2	...
3	...
10	...
...	

Incoming file from Audio Processing

Web Frontend

Replayed Audio

Time and Key information are sent in batches to client

Incoming Power for keys are converted into duty cycles for PWM signal

Micro-controller waits to send signals to the Piano Interface

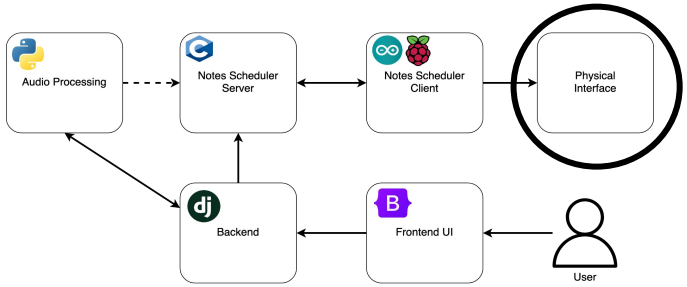
Piano Interface

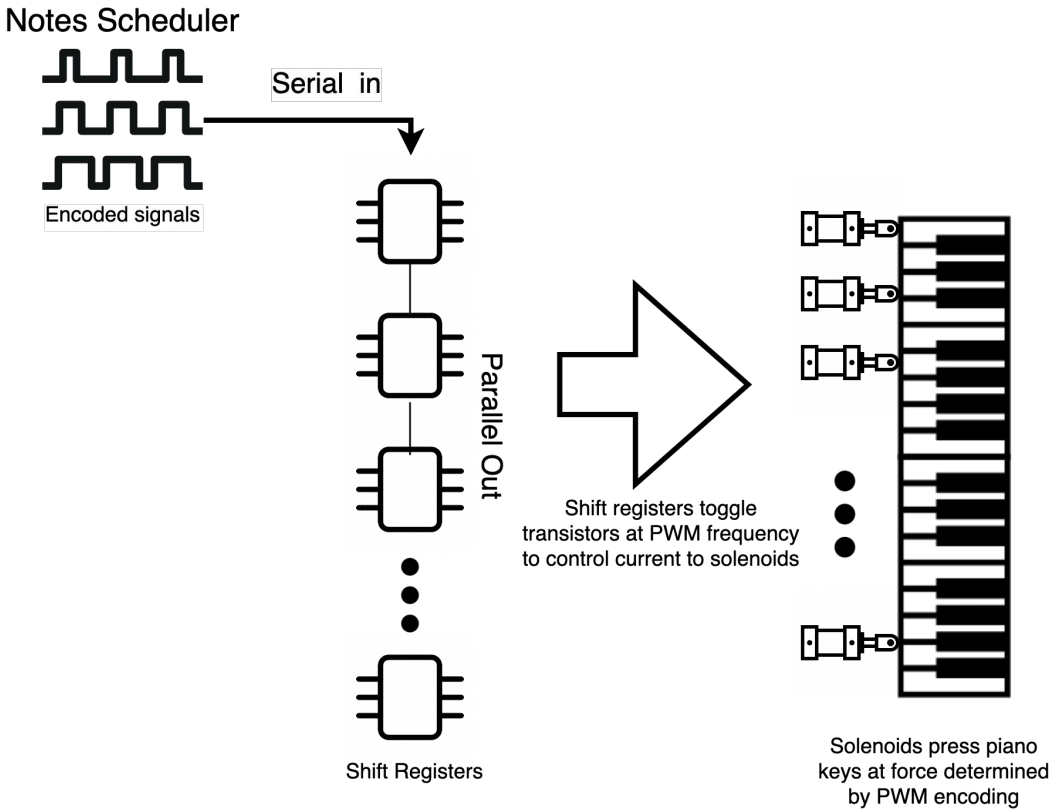
Playback controls

Server stops sending keys to the client

Server resumes sending keys to the client

Physical Interface





Risk Factors and Unknowns

- Building the physical interface
  - Might take longer than expected, therefore we'll build a proof of concept build that only uses ~5 solenoids to press keys
- Blurring the audio might not extract enough information
- Upload and Download internet speed between remote server and Raspberry Pi introduce a bottleneck
- Our piano play rate may be too high, causing keys to be "spammed"

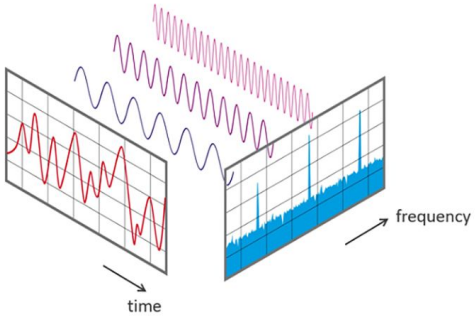
Alternative Design Strategies

- Virtual piano implementation: should our proof of concept for the piano-playing mechanism fail, we will implement a virtual piano solution.
- Near real-time speech-to-piano translation: once MVP is achieved, we hope to allow people to speak and hear their sentence played on the piano once they’re done speaking.
- Lower-latency backend: once fully committed to the physical interface, we can migrate the backend logic onto a Jetson if speed is a concern.

Testing, Verification, Metrics

- Web-app physical system latency
  - Use Selenium to mimic clicking on UI components
  - Measure the time between pressing a button on our frontend UI and the appropriate reaction of the system
  - Our goal metric is < 200ms

- Fast Fourier transform accuracy
  - Use an input audio recording we create with known frequencies and amplitudes
    - For each window we will compare the frequencies reported by our system to the known frequencies at that time
  - Accuracy dependent on chosen time window for FFT
    - Shorter Window=> Less Accurate
    - Longer Window => Long wait times
  - Adjust our time window for >80% accuracy.

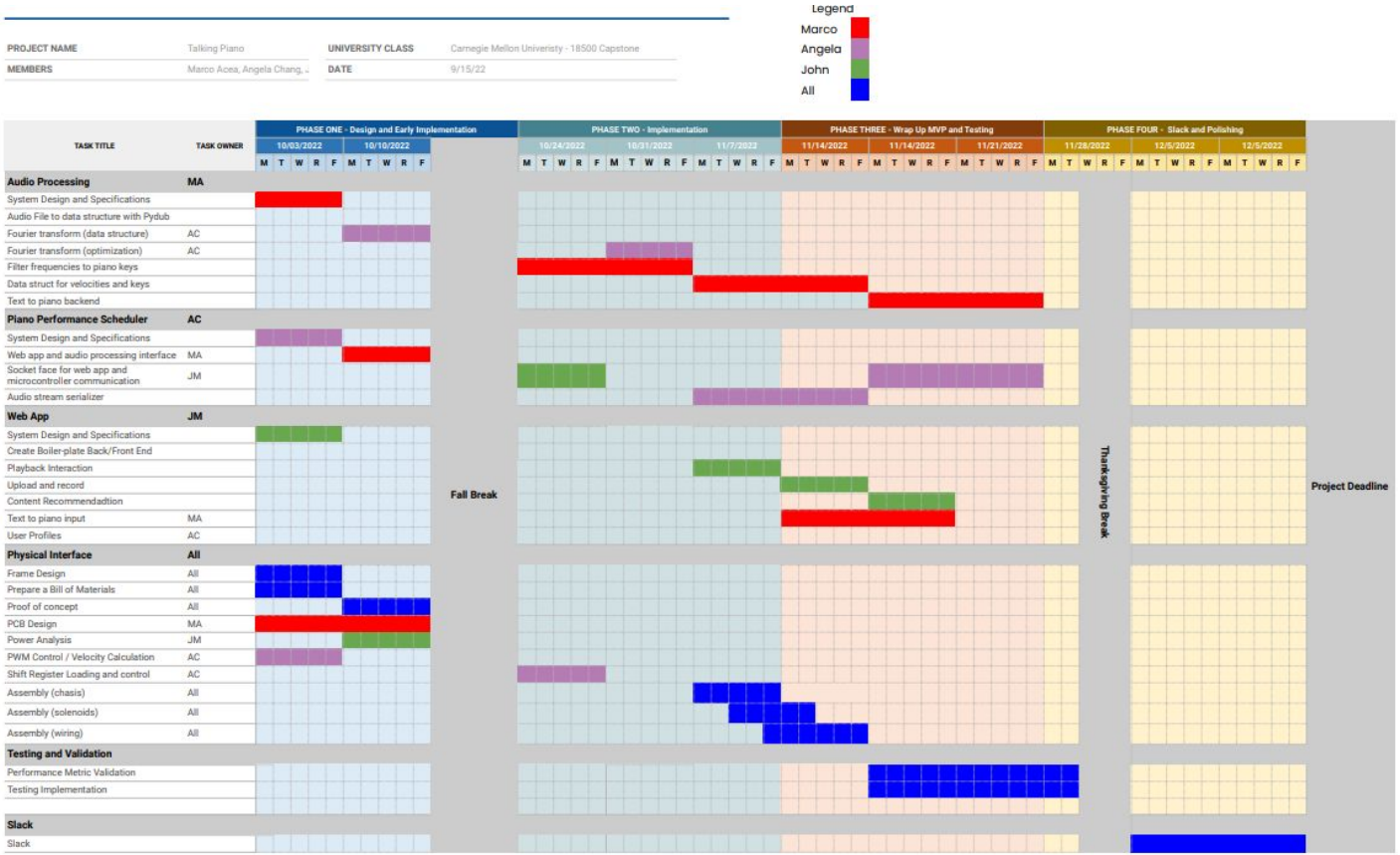


Testing, Verification, Metrics cont.

- Syllable timing
  - Use recordings with labeled start times for each syllable
  - Record the number of syllables whose start time does not match the original recordings labelled start time

- Fidelity of Output Audio
  - Generate a series of prompts and output piano recordings
  - Survey a group of listeners on whether or not they can understand the prompt given the piano audio
  - Collect data on what percentage of listeners were able to make out what the piano was trying to say

Gantt Chart


PROJECT NAME
Talking Piano
UNIVERSITY CLASS
Carnegie Mellon Univeristy - 18500 Capstone
MEMBERS
Marco Acea, Angela Chang, ..
DATE
9/15/22
Legend
Marco
Angela
John
All
TASK TITLE
TASK OWNER
PHASE ONE - Design and Early Implementation
PHASE TWO - Implementation
PHASE THREE - Wrap Up MVP and Testing
PHASE FOUR - Slack and Polishing
10/03/2022
10/10/2022
10/24/2022
10/31/2022
11/7/2022
11/14/2022
11/14/2022
11/21/2022
11/28/2022
12/5/2022
12/5/2022
M T W R F M T W R F M T W R F M T W R F M T W R F M T W R F M T W R F M T W R F M T W R F M T W R F M T W R F
Audio Processing MA
System Design and Specifications
Audio File to data structure with Pydub
Fourier transform (data structure) AC
Fourier transform (optimization) AC
Filter frequencies to piano keys
Data struct for velocities and keys
Text to piano backend
Piano Performance Scheduler AC
System Design and Specifications
Web app and audio processing interface MA
Socket face for web app and microcontroller communication JM
Audio stream serializer
Web App JM
System Design and Specifications
Create Boiler-plate Back/Front End
Playback Interaction
Upload and record
Content Recommendadtion
Text to piano input MA
User Profiles AC
Physical Interface All
Frame Design All
Prepare a Bill of Materials All
Proof of concept All
PCB Design MA
Power Analysis JM
PWM Control / Velocity Calculation AC
Shift Register Loading and control AC
Assembly (chasis) All
Assembly (solenoids) All
Assembly (wiring) All
Testing and Validation
Performance Metric Validation
Testing Implementation
Slack
Slack
Fall Break
Thanksgiving Break
Project Deadline