INSTRUMENT RECOGNITION

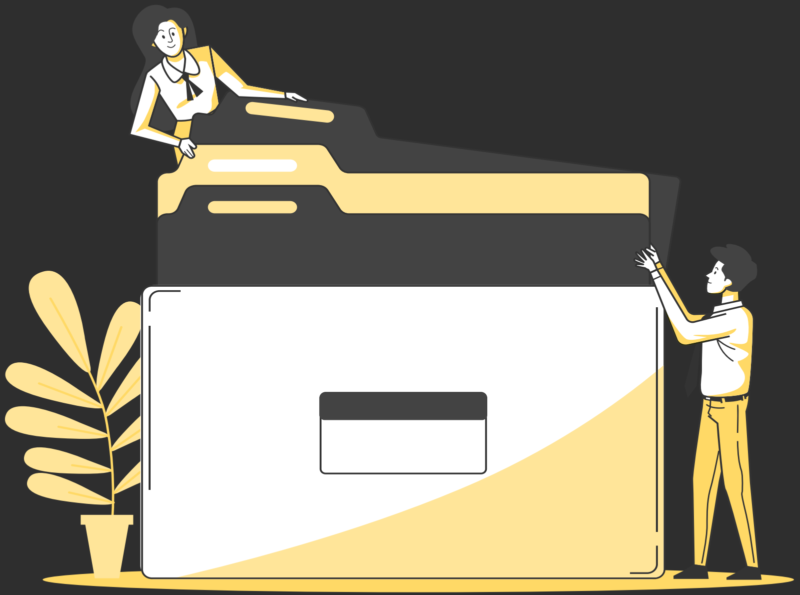
Aleks | Aner | Axel | Cuong | Joe | Thomas

Business Requirements Document

- We did not have any significant updates in our BRD

Management Plan

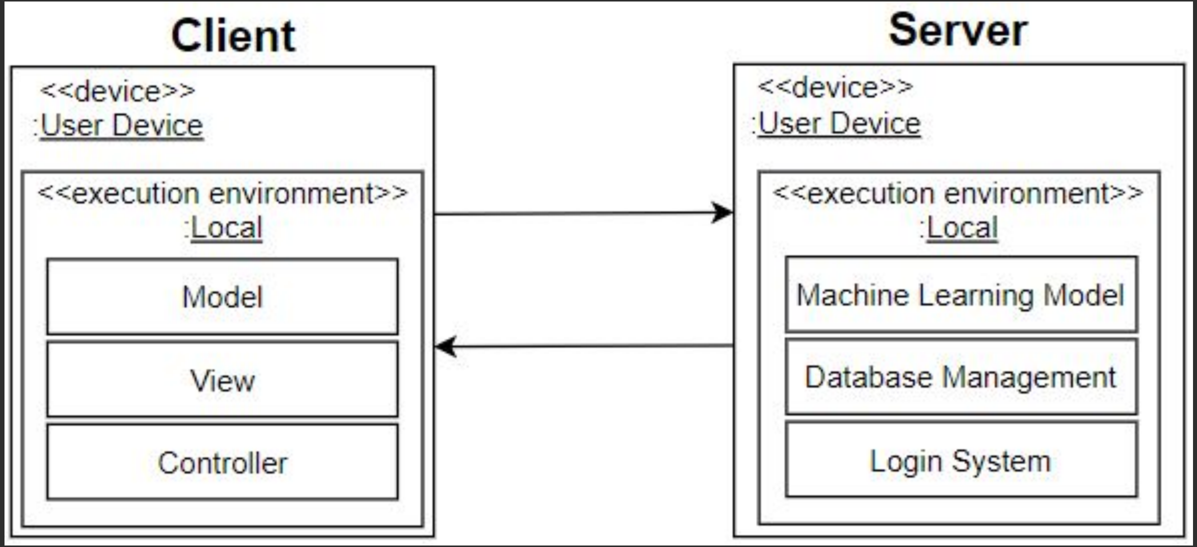
- We had updated our Sprint Board and our Burndown Chart



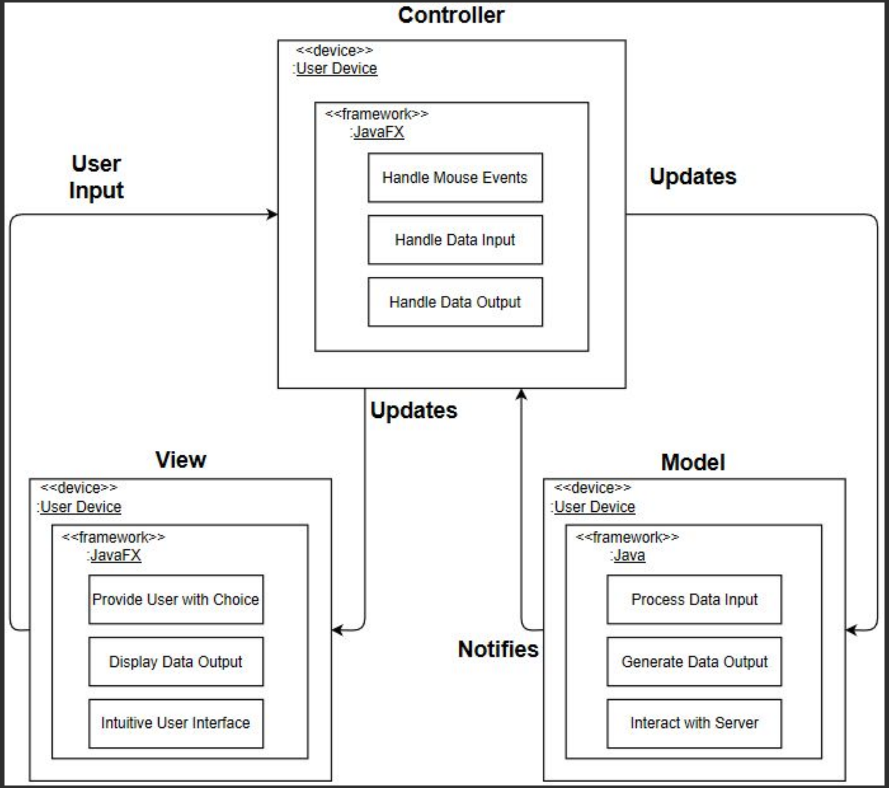
Architecture and Design

- We had slight updates to the Architecture and Design since last sprint

Client and Server Communication



Model View Controller Architecture



Controller

<<device>>
:User Device

<<framework>>
:JavaFX

Handle Mouse Events

Handle Data Input

Handle Data Output

User Input

Updates

Updates

Notifies

View

<<device>>
:User Device

<<framework>>
:JavaFX

Provide User with Choice

Display Data Output

Intuitive User Interface

Model

<<device>>
:User Device

<<framework>>
:Java

Process Data Input

Generate Data Output

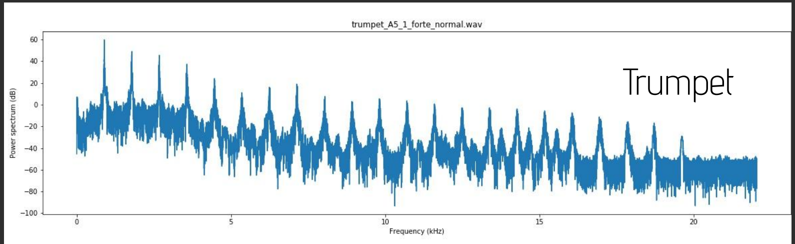
Interact with Server

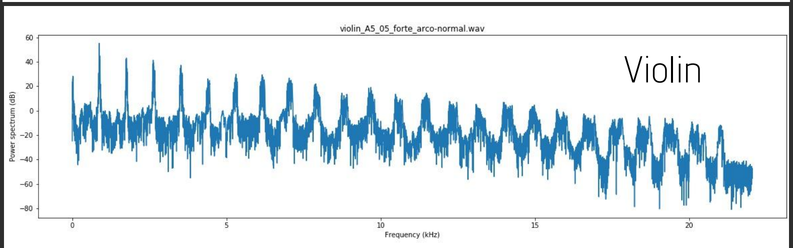
Our input/output data

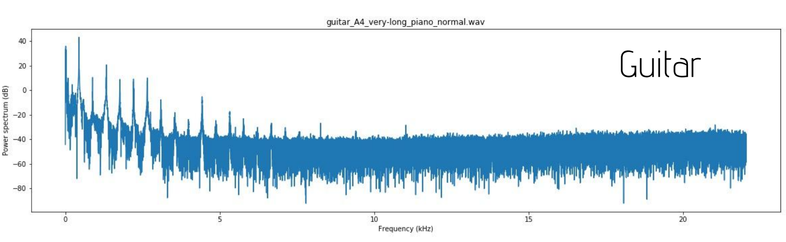
- X = .wav graphs
- Y = String/int labels

- Sets of 1,000 for training data

- Takes input graphs and outputs integer labels that can be decoded to Strings of instruments

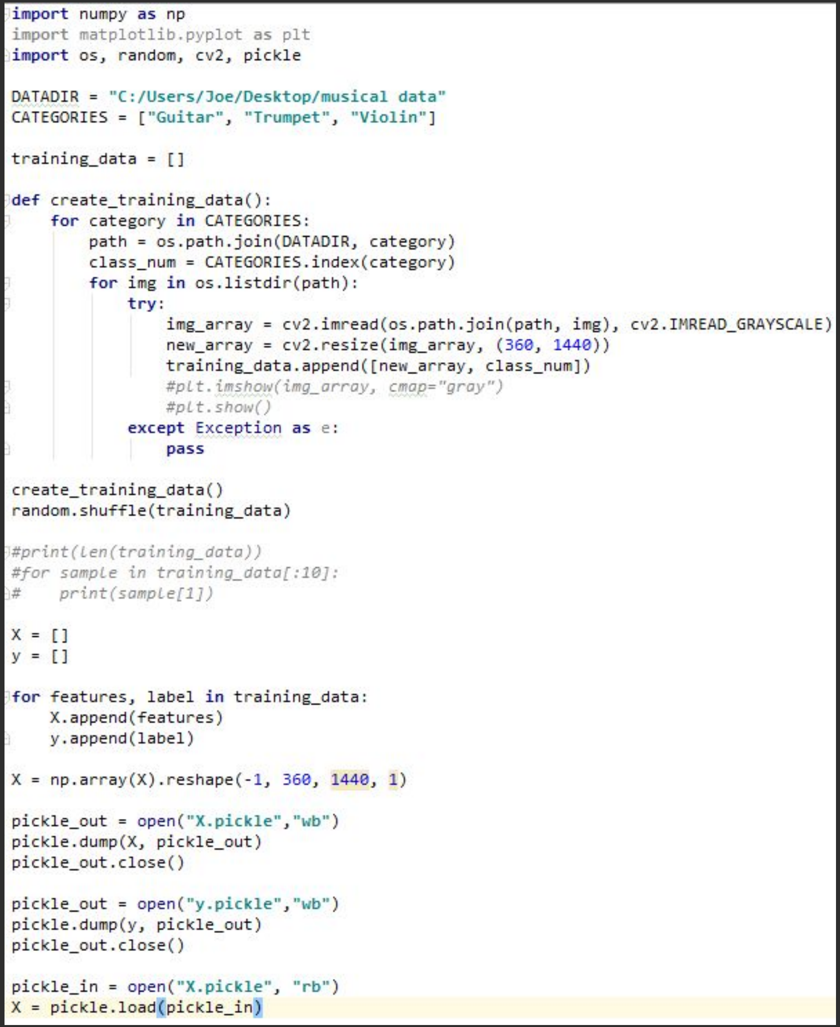






Importing our data

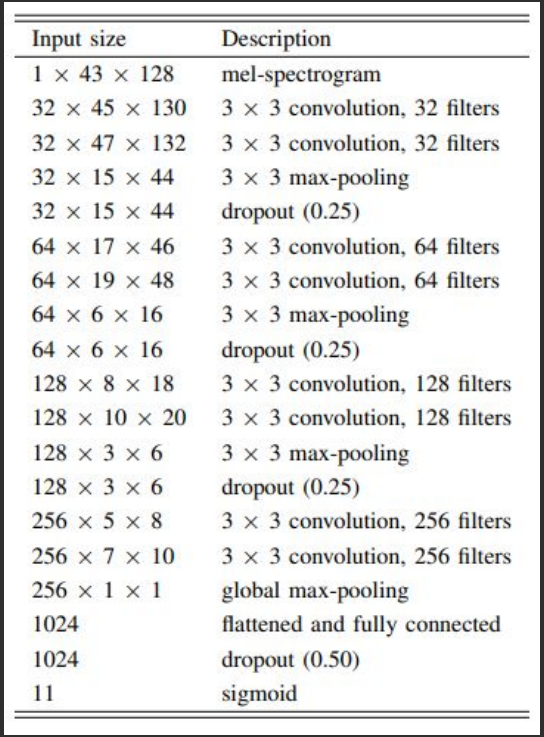
- Local file upload vs Google Colab upload
- Google Colab Issues
    - No referenceable local data
    - Re-uploading to Google Drive
    - Image unreadable
    - Training set storage
    - Little to no learning references



```python
import numpy as np
import matplotlib.pyplot as plt
import os, random, cv2, pickle

DATADIR = "C:/Users/Joe/Desktop/musical data"
CATEGORIES = ["Guitar", "Trumpet", "Violin"]

training_data = []

def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (360, 1440))
                training_data.append([new_array, class_num])
                #plt.imshow(img_array, cmap="gray")
                #plt.show()
            except Exception as e:
                pass

create_training_data()
random.shuffle(training_data)

#print(len(training_data))
#for sample in training_data[:10]:
#    print(sample[1])

X = []
y = []

for features, label in training_data:
    X.append(features)
    y.append(label)

X = np.array(X).reshape(-1, 360, 1440, 1)

pickle_out = open("X.pickle","wb")
pickle.dump(X, pickle_out)
pickle_out.close()

pickle_out = open("y.pickle","wb")
pickle.dump(y, pickle_out)
pickle_out.close()

pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in)
```

Layers

- 1 x 360 x 1440 (Input)
- 32 x 356 x 1436 5 x 5 s = 1 (Conv)
- Math: (360-5)/1 + 1 = 356
- 32 x 352 x 1432 5 x 5 s = 1 (Conv)
- 32 x 176 x 716 2 x 2 s = 2 (Pool)
- Math: (352-2)/2 + 1 = 176
- 32 x 176 x 716 Dropout (0.25)
- 64 x 172 x 712 5 x 5 s = 1 (Conv)
- 64 x 168 x 708 5 x 5 s = 1 (Conv)
- 64 x 84 x 354 2 x 2 s = 2 (Pool)
- 64 x 84 x 354 Dropout (0.25)
- Continued: 84 -> 38 -> 15



Input size	Description
1 × 43 × 128	mel-spectrogram
32 × 45 × 130	3 × 3 convolution, 32 filters
32 × 47 × 132	3 × 3 convolution, 32 filters
32 × 15 × 44	3 × 3 max-pooling
32 × 15 × 44	dropout (0.25)
64 × 17 × 46	3 × 3 convolution, 64 filters
64 × 19 × 48	3 × 3 convolution, 64 filters
64 × 6 × 16	3 × 3 max-pooling
64 × 6 × 16	dropout (0.25)
128 × 8 × 18	3 × 3 convolution, 128 filters
128 × 10 × 20	3 × 3 convolution, 128 filters
128 × 3 × 6	3 × 3 max-pooling
128 × 3 × 6	dropout (0.25)
256 × 5 × 8	3 × 3 convolution, 256 filters
256 × 7 × 10	3 × 3 convolution, 256 filters
256 × 1 × 1	global max-pooling
1024	flattened and fully connected
1024	dropout (0.50)
11	sigmoid

MFCC -> PSD
43x128 -> 360x1440

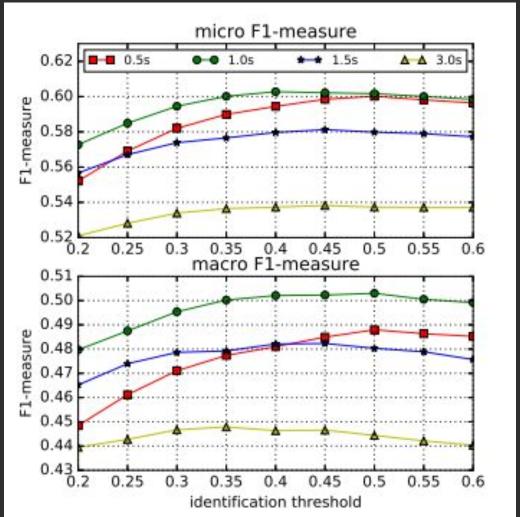
Total: 1 + 4 x 4 + 1 + 1 = 19

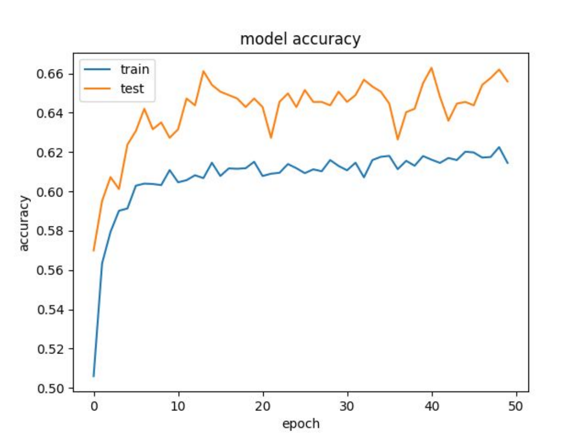
Accuracy

- Preliminary peer models ranked roughly 40-60% for 4 labels
- Initial accuracy tracking at 60-70% for 3 labels

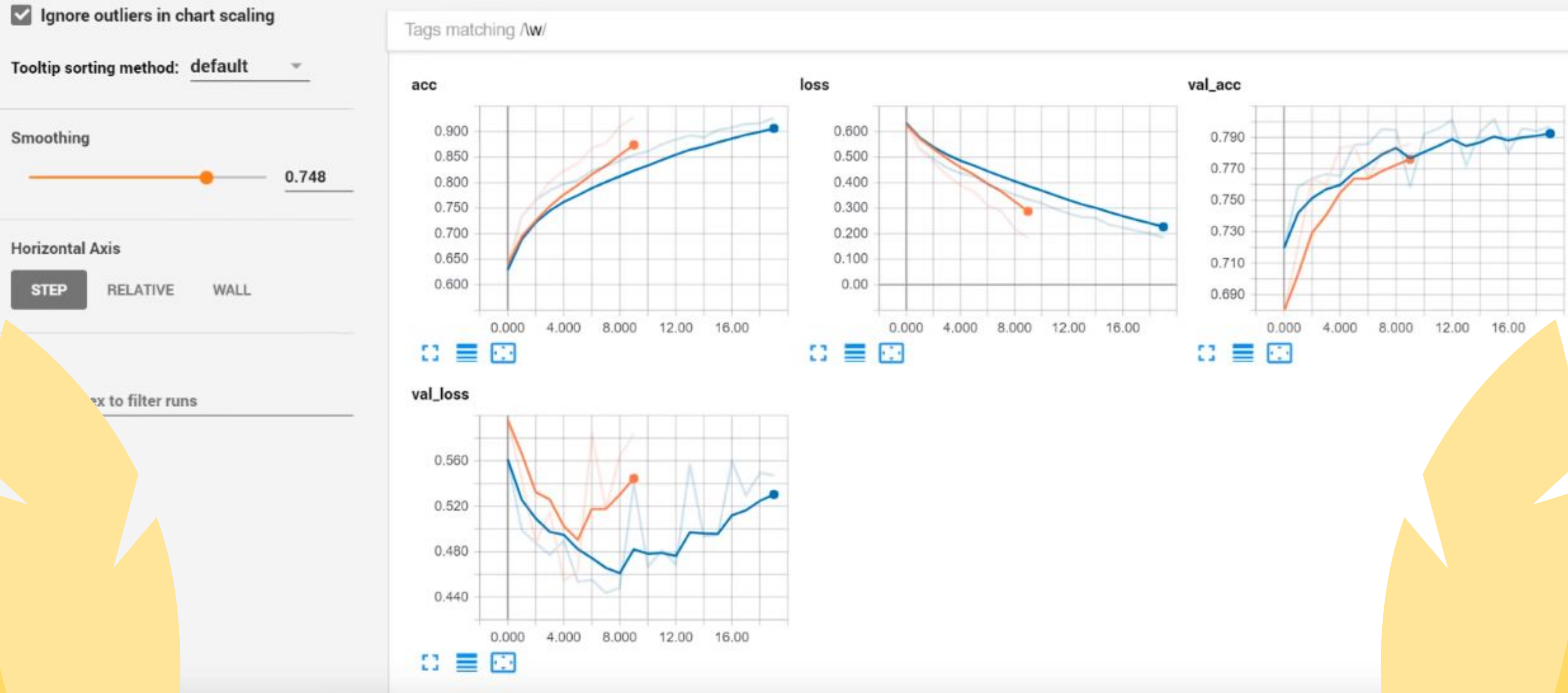
- Accuracy chart shows underfitting by 6-10%
- Little improvement beyond the 14th epoch

- Accuracy thresholds:
  - Functional: 70%
  - Commercial 90%





Data Visualization using TensorBoard



☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.748

Horizontal Axis

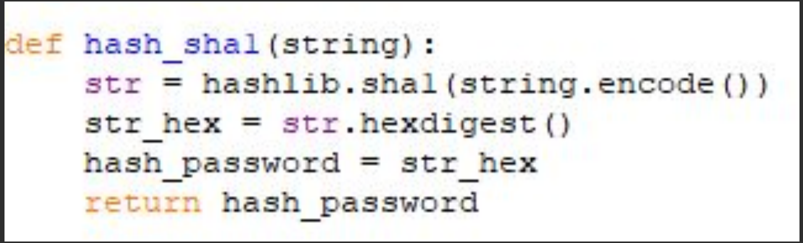
STEP RELATIVE WALL

ex to filter runs

Tags matching /\w/

Login Hash

- Passwords tend to be easily attacked and hacked with brute force or a dictionary attack.
- What hashing will do is it will rewrite the password as random characters to make it hard to guess.
- Secure Hash Algorithm 1 is a cryptographic hash function which takes input and produces a 160 - bit (20 - byte) hash value known as a message digest - typically rendered as a hexadecimal number, 40 digits long.



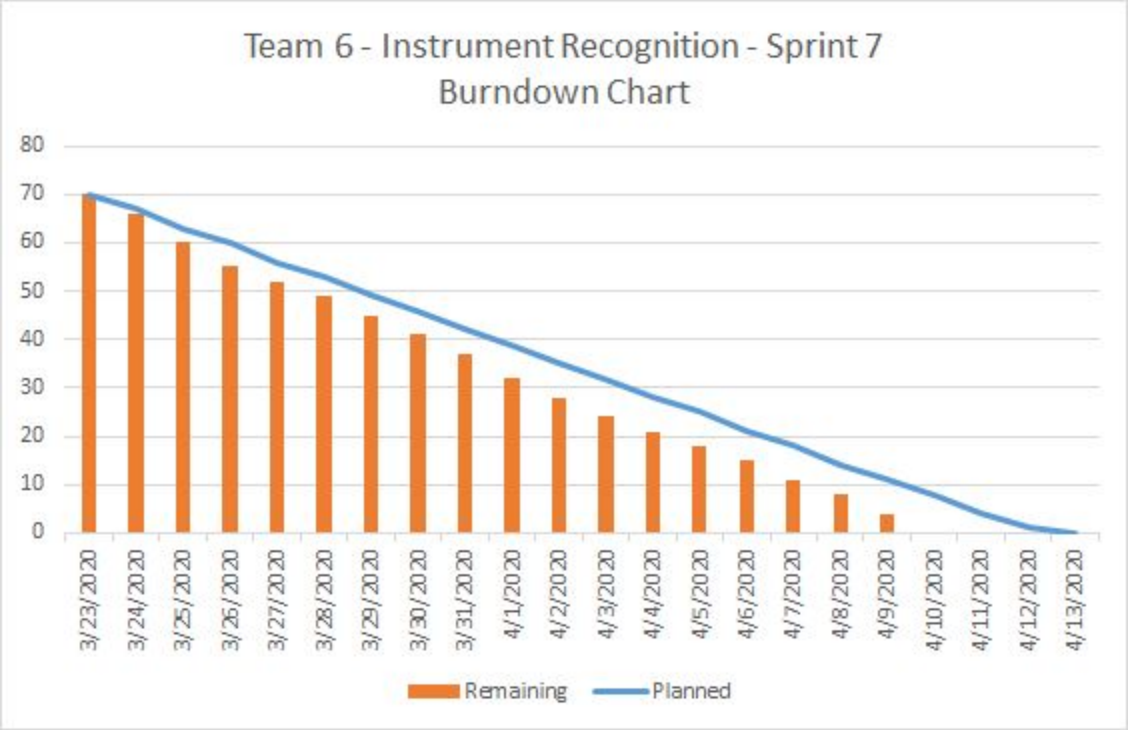
```
def hash_sha1(string):
    str = hashlib.sha1(string.encode())
    str_hex = str.hexdigest()
    hash_password = str_hex
    return hash_password
```

Demo

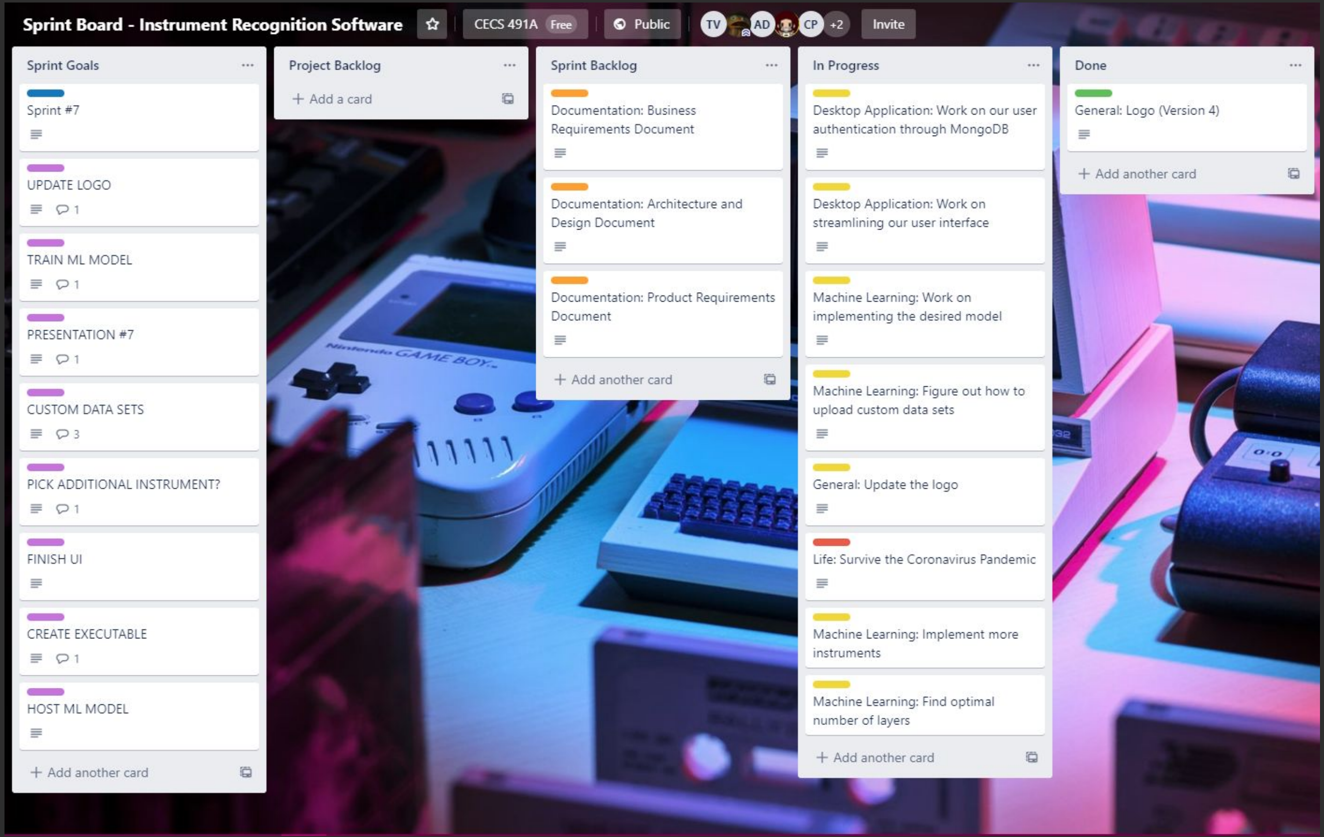
Sprint 7 Retrospective

- Sprint Goals
  - Develop an executable file
  - Create a functional monophonic model
- Points
  - Planned: 60
  - Achieved: 70
- Things to improve:
  - In client digestion of audio into graphs
  - Streamline UI
  - Monophonic expansion of instruments
  - Polyphonic implementation
  - Data projection to UI display

Burndown Chart



Sprint Board



Sprint Board - Instrument Recognition Software

Sprint Goals

- Sprint #7
- UPDATE LOGO 💬 1
- TRAIN ML MODEL 💬 1
- PRESENTATION #7 💬 1
- CUSTOM DATA SETS 💬 3
- PICK ADDITIONAL INSTRUMENT? 💬 1
- FINISH UI
- CREATE EXECUTABLE 💬 1
- HOST ML MODEL

Project Backlog

Sprint Backlog

- Documentation: Business Requirements Document
- Documentation: Architecture and Design Document
- Documentation: Product Requirements Document

In Progress

- Desktop Application: Work on our user authentication through MongoDB
- Desktop Application: Work on streamlining our user interface
- Machine Learning: Work on implementing the desired model
- Machine Learning: Figure out how to upload custom data sets
- General: Update the logo
- Life: Survive the Coronavirus Pandemic
- Machine Learning: Implement more instruments
- Machine Learning: Find optimal number of layers

Done

- General: Logo (Version 4)

Thanks for listening!