# Instrument Recognition Software

Aleks
Aner
Axel
Cuong
Joe
Thomas

# Business Requirements Document

- We did not have any significant update in our BRD

# Management Plan

- We have updated our Sprint Board and Burndown Chart

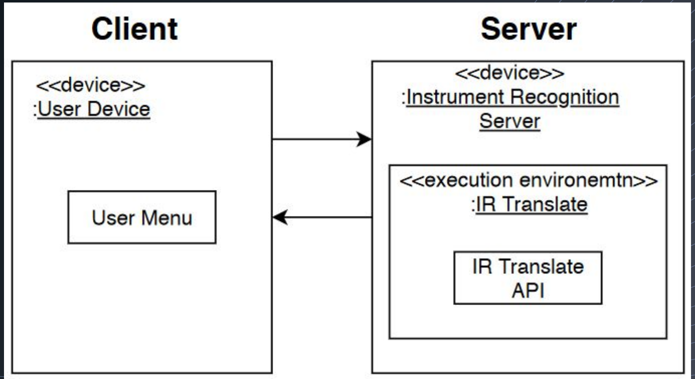
# Architecture and Design (Update)

- Included our primary data source
  - Philharmonia Orchestra
    - URL: philharmonia.co.uk/explore/sound_samples
    - Provides an adequate amount of audio recordings that we will use as our main dataset source for training and testing
    - The audio files are in a lossy MPEG Audio Layer-3 format (MP3), which we will be converting into a lossless Waveform audio file format (WAV)
      - This conversion will enhance the clarity and readability of the audio file, resulting in a more accurate sound data.
      - We also aim to have a built-in MP3 to WAV in our program.
    - There are a total of 57 instruments, each with an adequate amount of samples with varying pitches, lengths, dynamics, and articulation.
    - We are planning on only using a small fraction of the samples in our project (around two to three instruments)

# Architecture

- Client
  - The user's side of the software that contains a menu which allows them to interact with the software

- Server
  - The server handles the user registration and login authentication, machine learning model, and handles the calculation of the algorithms.



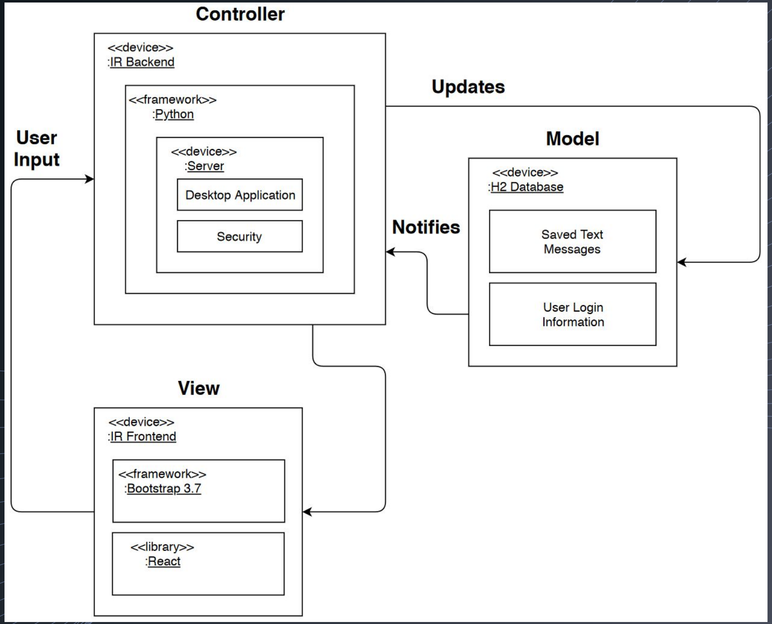
Client-Server interaction

# Architecture

- Controller
  - The controller manages the updates and serves as the main process for Model and View

- Model
  - The Model contains the login information and audio files

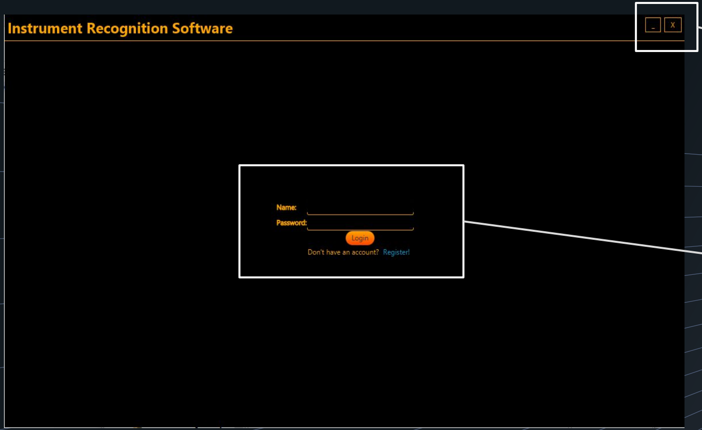
- View
  - The View provides the user with the final outcome



Controller-View-Model interaction

# Architecture

- Front-end
  - JavaFX
    - Built using JavaFX SDK
    - Desktop application

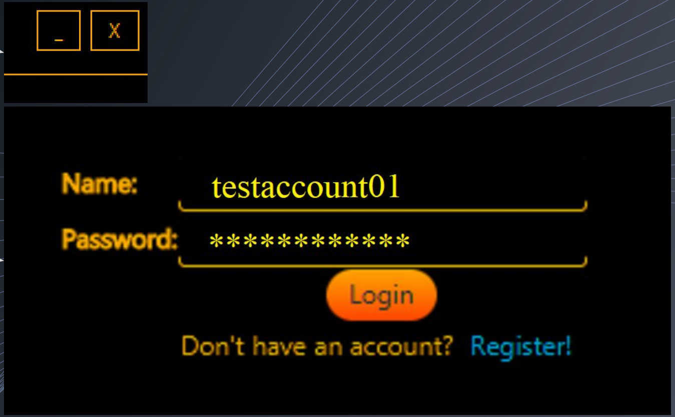


Instrument Recognition Software

Name:

Password:

Prototype registration and login page



Name: testaccount01

Password: ************

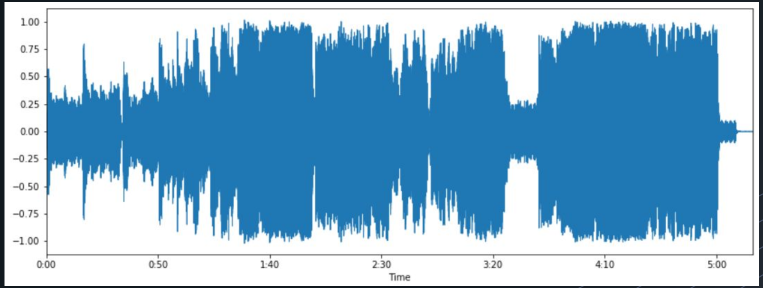
Prototype registration and login page

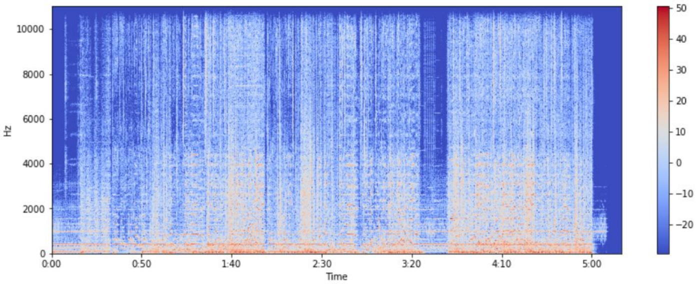
# Architecture

- Back-end
    - Server language: Python
    - Database: MongoDB (non-relational)
        - Connected using PyMongo
        - Stores the usernames and hashed passwords

# File to Data Input

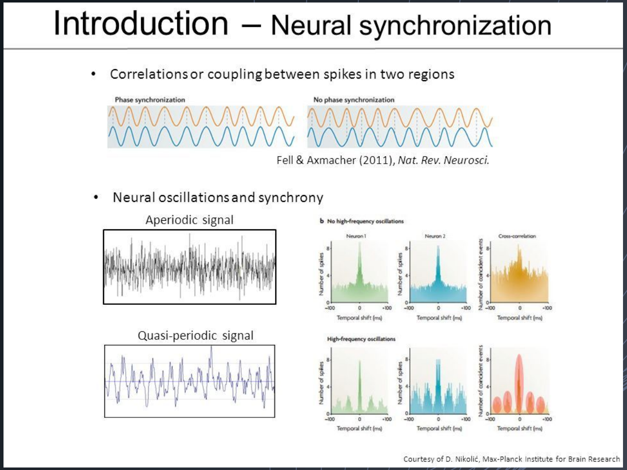
- In Python
    - Using LibROSA library
        - Have LibROSA analyze the file and output a stereo waveform and linear power spectrogram of the file data
        - Use Keras for machine learning in analysis and classification of the outputs from LibROSA





# File to Data Input

- Current input: (60000, 28, 28, 1)
    - (batch size, width, hight, channels)
- Final input: (training size, Xmax, Xmax, 1)
    - We cannot know the explicit input data to our function at the given time but can point to the factors that will define it



## Introduction – Neural synchronization

- Correlations or coupling between spikes in two regions

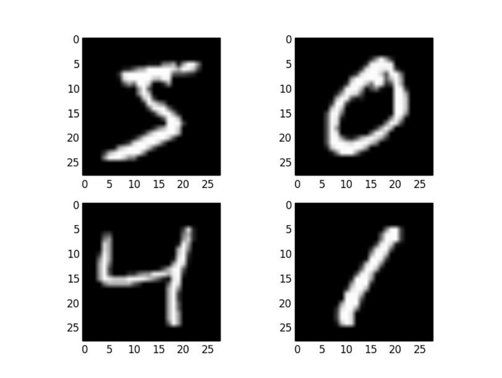
Phase synchronization
No phase synchronization

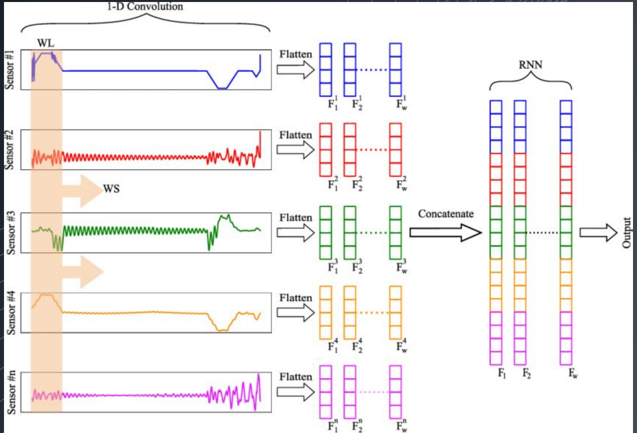
Fell & Axmacher (2011), *Nat. Rev. Neurosci.*

- Neural oscillations and synchrony

Aperiodic signal
Quasi-periodic signal
b No high-frequency oscillations
High-frequency oscillations
Neuron 1
Neuron 2
Cross-correlation
Number of spikes
Number of coincident events
0
4
8
−100
0
+100
Temporal shift (ms)

Courtesy of D. Nikolić, Max-Planck Institute for Brain Research





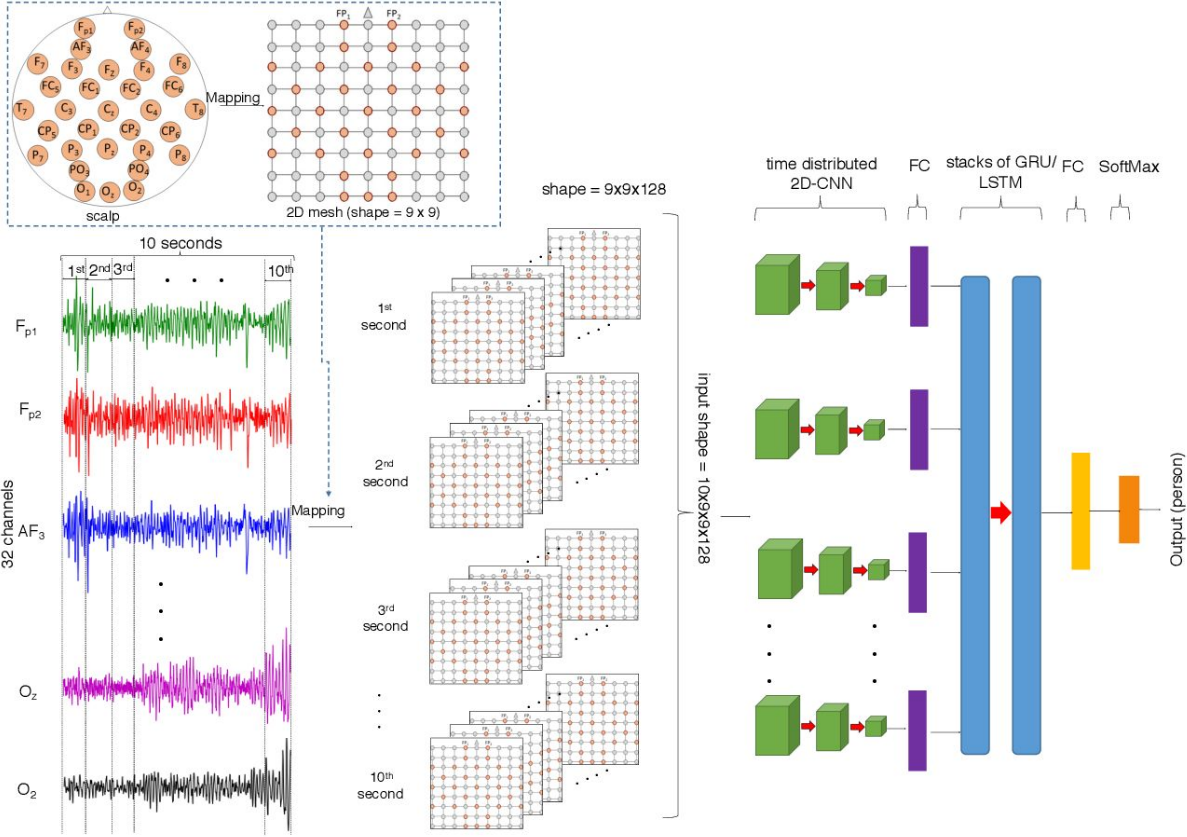


Figure 1: A CNN graphic of EEG PI (Electroencephalography Personal Identification) from ArXiv

# Neural Network Layers

Anticipated number of layers: 15-20

- IRJET Oxford model (19)
  - 3x3 filters
  - 2x2 max pooling

Deep neural network layers:

- Input Layer
- Hidden Layers
- Output Layer

"In general, you cannot analytically calculate the number of layers or the number of nodes to use per layer in an artificial neural network to address a specific real-world predictive modeling problem." - Jason Brownlee, Ph. D.

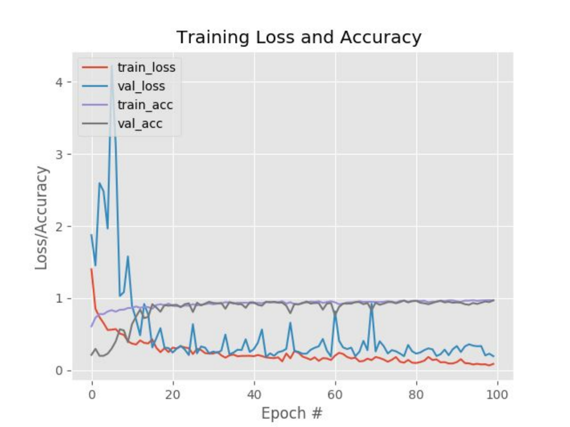
# Model Accuracy and Outputs

- Proof of Concept Model
  - 10k samples given 10 trials
  - Accuracy: 99.13%
  - Output: numeric labels

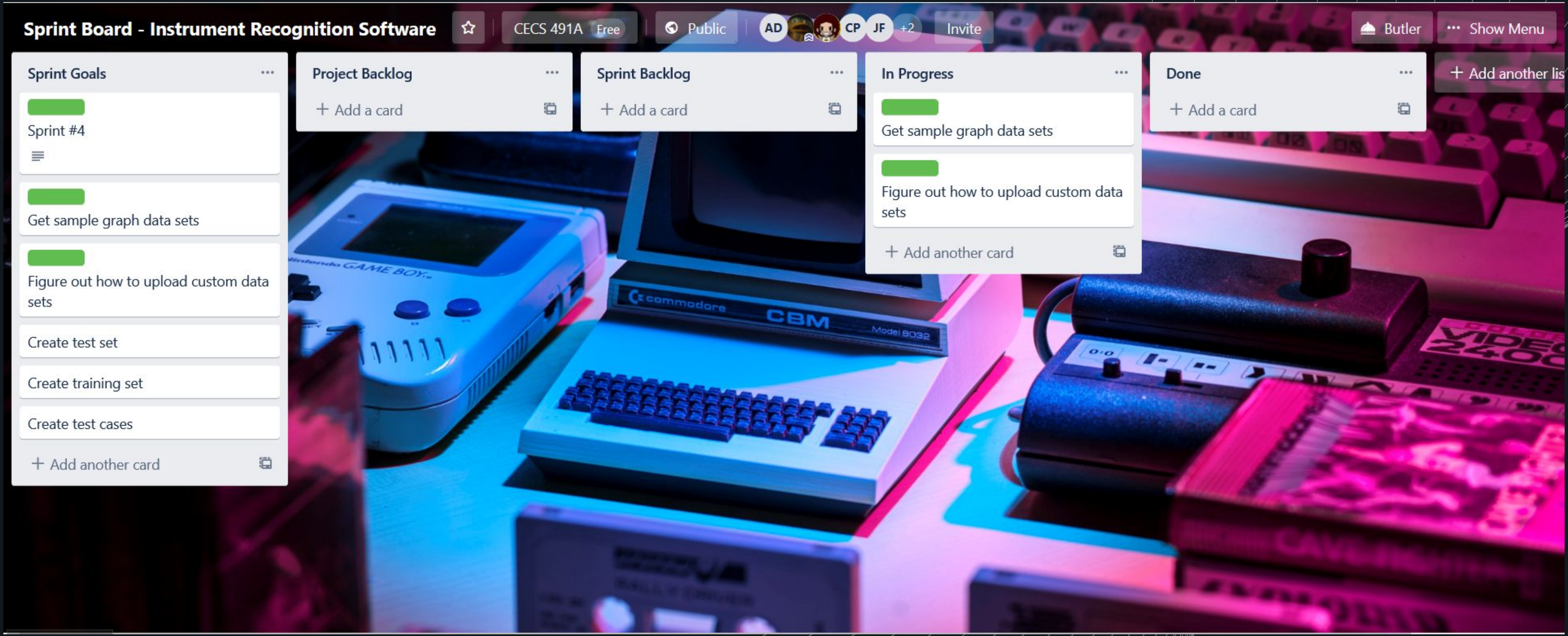
- Musical Recognition Model
  - Desired Accuracy: +90%
  - Acceptable Accuracy: +70%
  - Output: instrument labels

Over and Under fitting for PoC Model:

- 98.2% classification accuracy shows very minimal overfitting of data



# Sprint Retrospective



**Sprint Board - Instrument Recognition Software**

**Sprint Goals**

- 🟩 Sprint #4
- 🟩 Get sample graph data sets
- 🟩 Figure out how to upload custom data sets
- Create test set
- Create training set
- Create test cases

**Project Backlog**

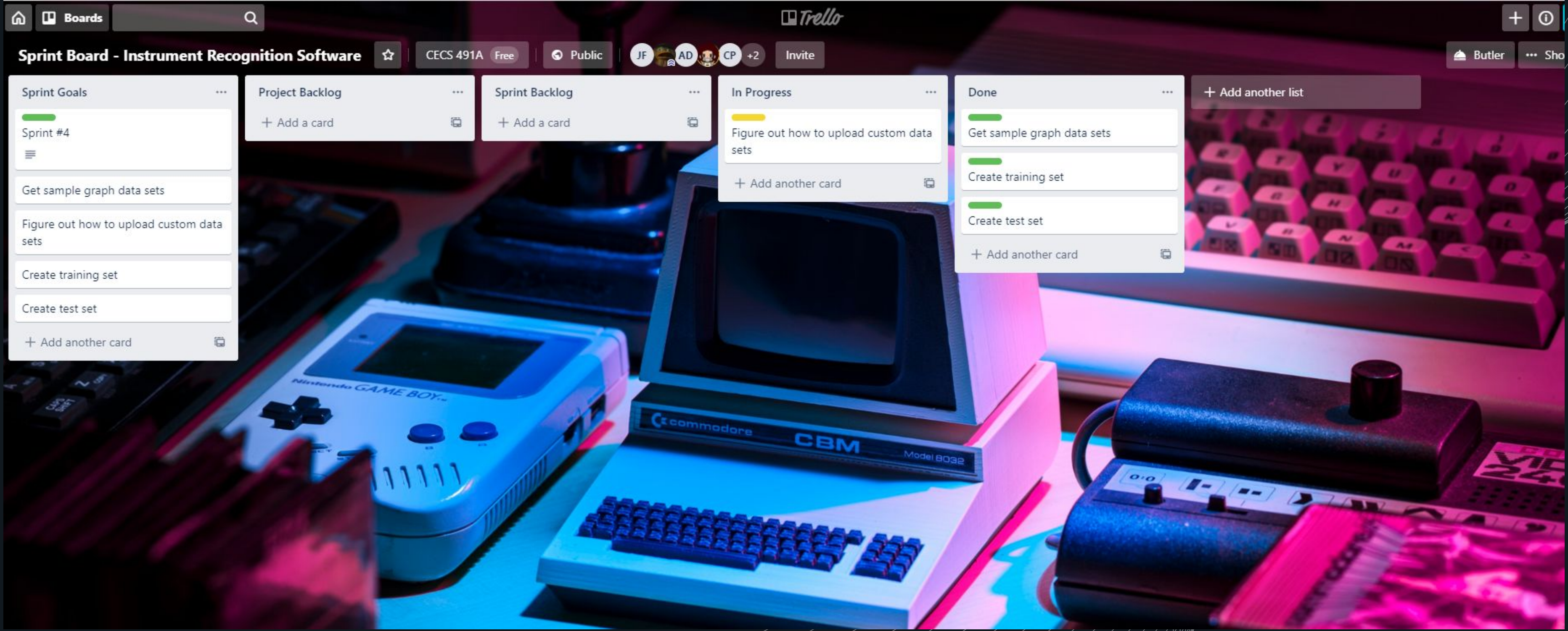
**Sprint Backlog**

**In Progress**

- 🟩 Get sample graph data sets
- 🟩 Figure out how to upload custom data sets

**Done**

Sprint Board at the beginning of the sprint



Sprint Board - Instrument Recognition Software

Sprint Goals
- Sprint #4
- Get sample graph data sets
- Figure out how to upload custom data sets
- Create training set
- Create test set
- Add another card

Project Backlog
- Add a card

Sprint Backlog
- Add a card

In Progress
- Figure out how to upload custom data sets
- Add another card

Done
- Get sample graph data sets
- Create training set
- Create test set
- Add another card

Sprint Board at the end of the sprint

# Sprint Summary

Sprint Goal

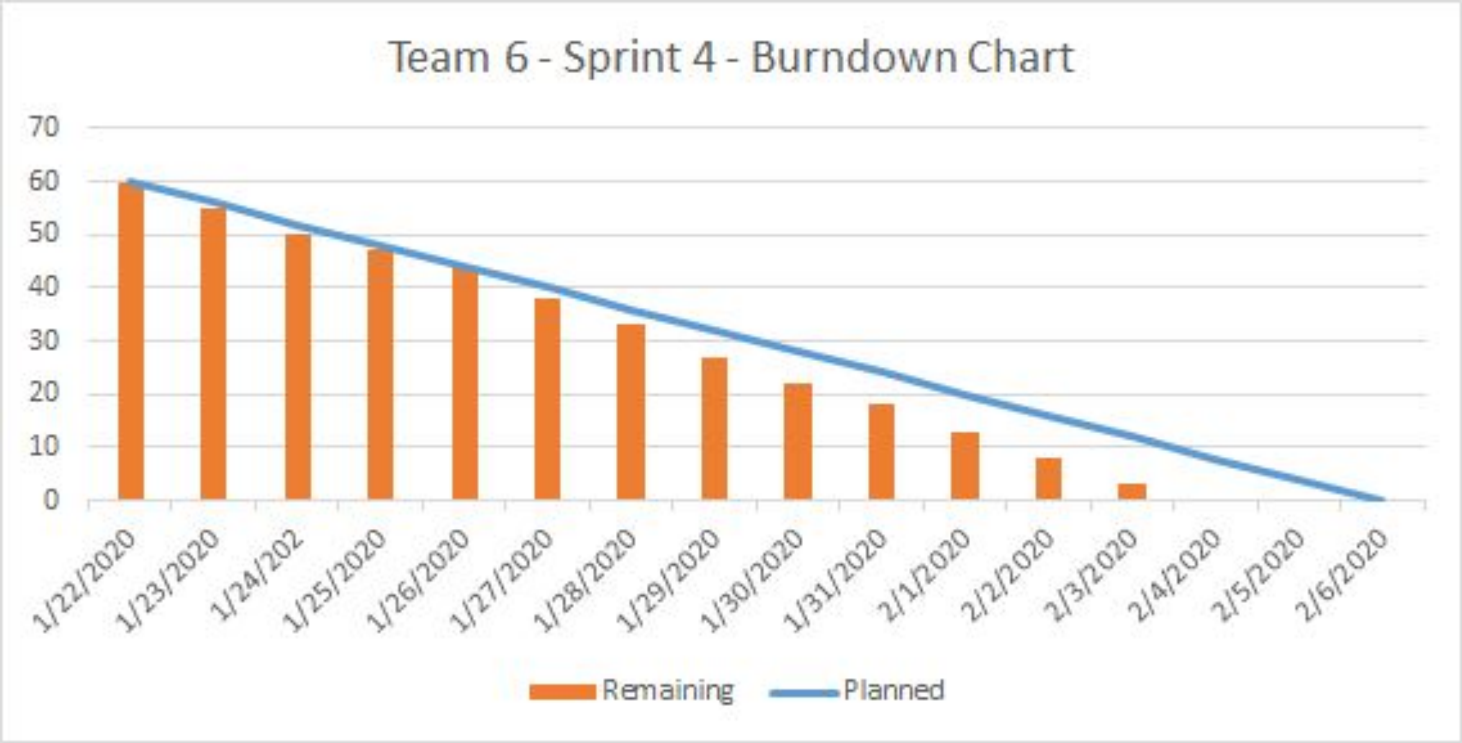
- Our main goal for this sprint was to develop testing and training datasets
- Determine a way CNN could interpret sound graphs
- Create a draft for CNN architecture

User Story Points

- Planned: 4 hours x 2.5 weeks x 6 members = 60 hours
- Achieved: 60

Next steps

- Run sample data through a CNN with 3 layers and 3 labels then test for accuracy
    - Projected completion: Mid March



Team 6 - Sprint 4 - Burndown Chart

# Thank You