Acapella

Team D5

Remote Sound Recording

Application area

- Free, easy to use web app
- Allows user to collaborate with ensemble members remotely
- Takes care of latency and audio delay

Django & Redis servers with peer-to-peer connection

Solution Approach

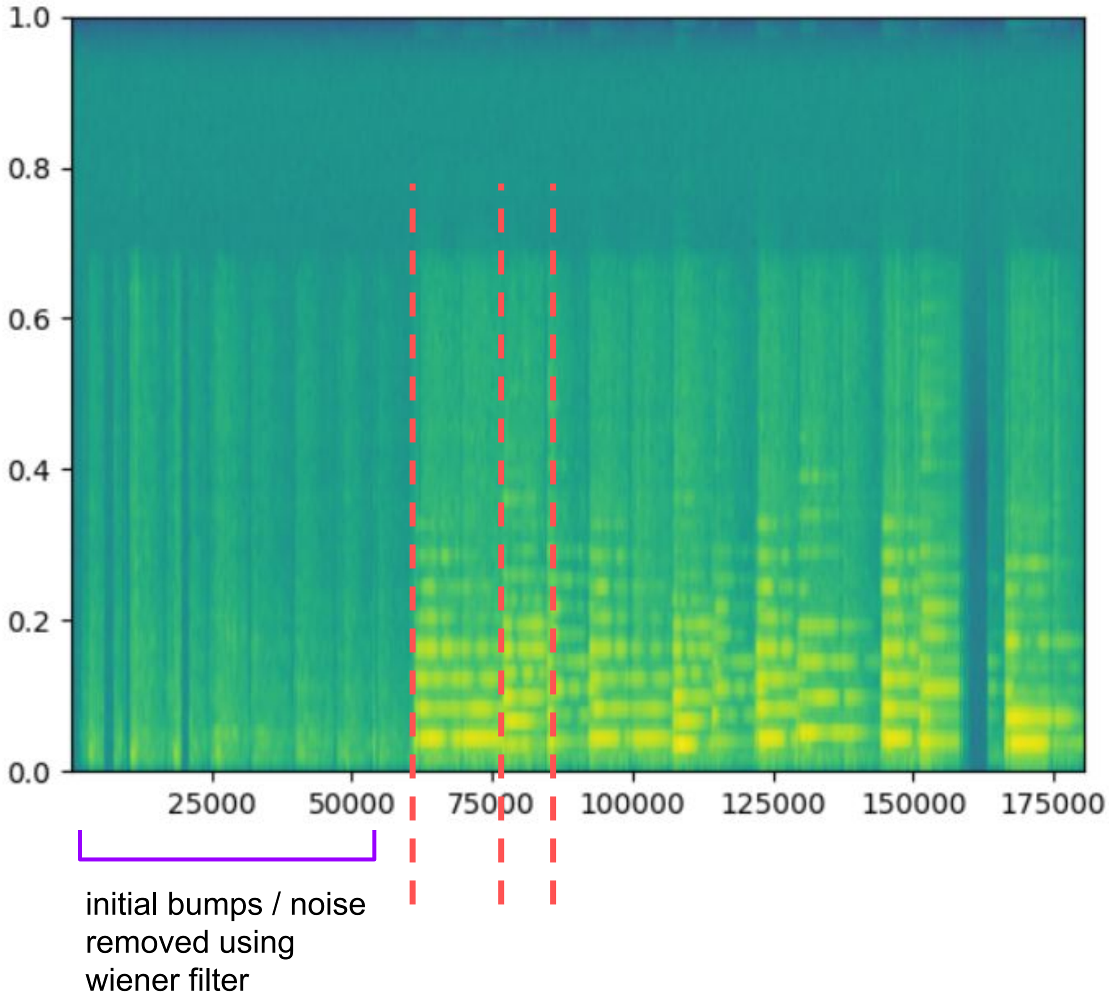
- Django - simple web server operations
  - Manages URLs and handles HTTP requests
  - Stores HQ audio on the server

- Redis - asynchronous WebSocket interface
  - Signalling for peer-to-peer connections

- Peer-to-peer connection - listening to each other in real-time
  - Users send audio to each other via UDP, using WebRTC API
  - Minimal latency at the expense of quality

Server-side syncing

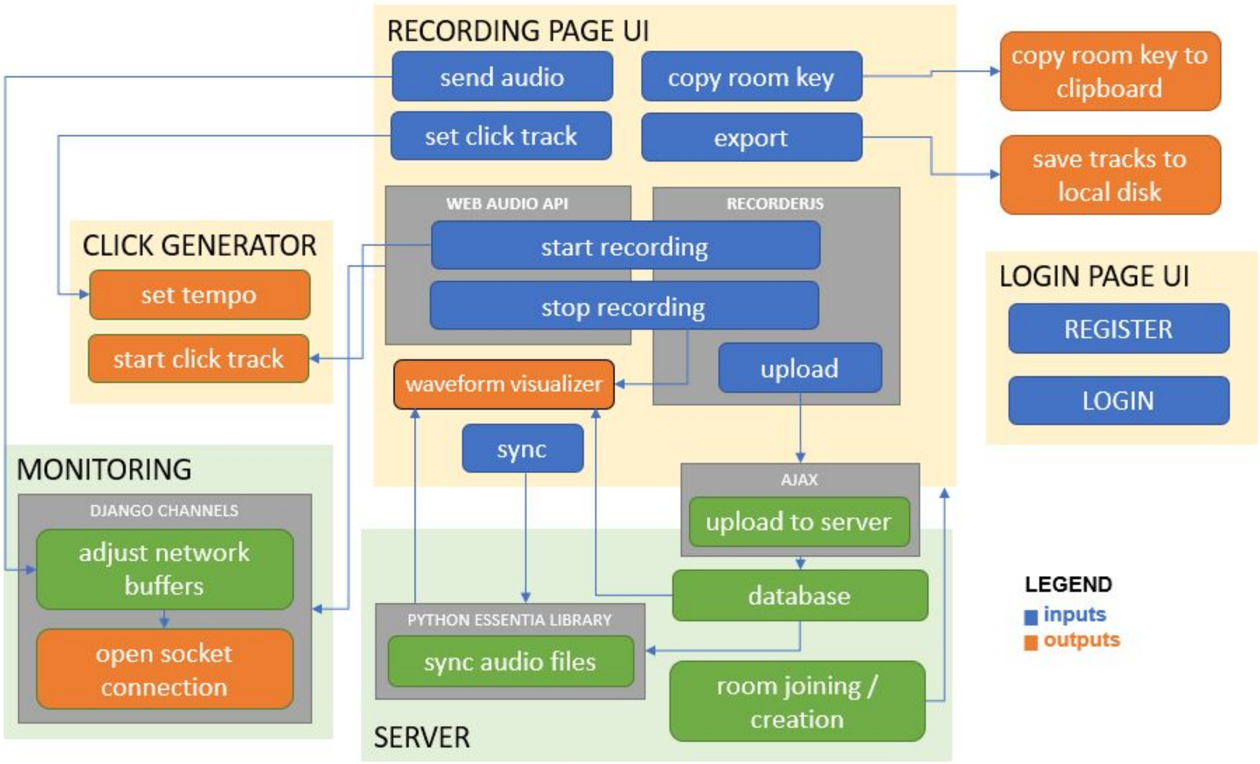
- get note beginning timea with onset detection
- Compare to array of beat times from click track
- Save new .wav file, adding or subtracting samples from beginning to match note onsets times with beat



System Diagram

Major changes

- Two web recorders
- Redis server for asynchronous WebSockets
- WebRTC for monitoring
- Separate upload and syncing



Establishing P2P Connection

Example: user A wants to connect to user B

- Each user generates their own SDP
  - Contains public-facing IP and port through which they can be connected to
- Handshaking:
  - User A sends an “offer” containing their SDP via WebSocket
  - User B sends an “answer” containing their SDP via WebSocket
- Connection can then be established independent of the server

P2P Connection

- Sending side:
  - Recorded audio is grouped into packets
  - Size of packet determined automatically to minimize latency

- Receiving side:
  - Received audio packets are placed into a jitter buffer for playback
  - Size of jitter buffer also determined automatically

- End-to-end latency is the time from when the audio is first grouped into packets by user A, to the time it is played back by user B

Design Trade-Offs

- .wav encoding
  - CD quality, lossless PCM encoding
  - More workable with python libraries
  - But no native support by Web Audio API
  - Solution: use Recorderjs, which exports recordings as .wav files

- **UDP** vs TCP
  - UDP: lower latency, but possibility of packet loss
  - Can use WebRTC, which maintains stable connection while prioritizing low latency

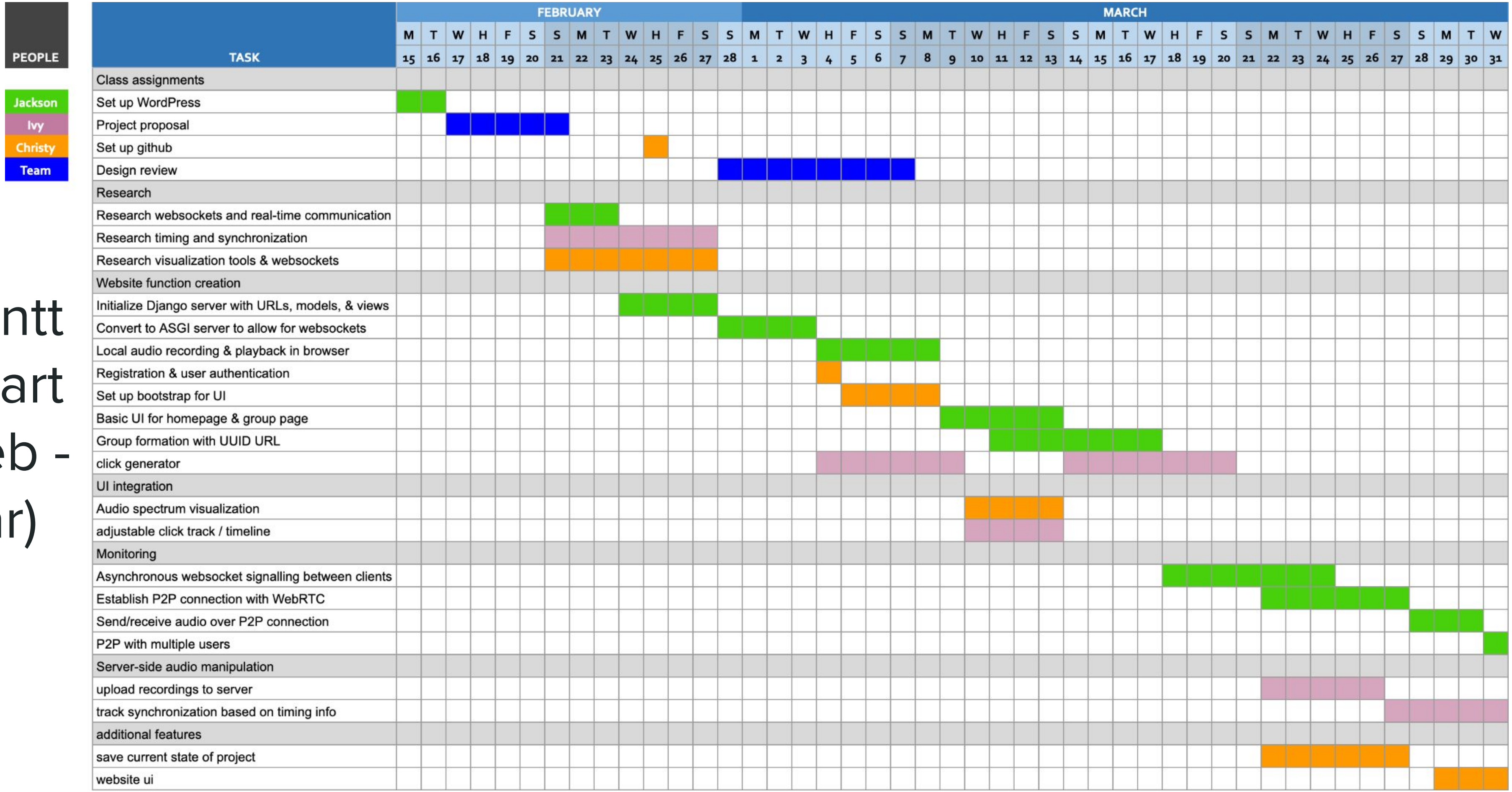
For the Public Demo

Complete Solution

- Go through all our features
  - Track ui
  - Peer-to-peer monitoring
  - Syncing
- Demo recording session
- Not yet done:
  - Cloud deployment, all testing that requires cloud deployment

METRIC	VALIDATION	PERFORMANCE
Latency < 100ms	Monitoring: Send time (UTC) with a packet once every 2 seconds and compare that to the UTC when it is received Synchronization: compare corresponding onset times of each of the uploaded tracks	Monitoring end-to-end latency: <5ms locally, TBD after cloud deployment Synchronization: 20ms
Audio quality < 5% packet loss	# lost packets / # sent packets	Packet-loss rate: virtually 0% locally, TBD after cloud deployment
UI intuitiveness < 5s to navigate	Poll a dozen users both familiar and unfamiliar with DAW interfaces, timing them on performing basic functions such as join room, create track, start recording etc.	TBD after cloud deployment
Comparative usefulness and avg satisfaction > 7	Survey users of our application, asking them to rate various functions, overall audio quality, and overall usefulness from a scale of 1-to-10	TBD after cloud deployment

Gantt Chart (Feb - Mar)



Gantt Chart (Apr - May)