# Intelligent Voice Recorder

Dhiraj Gurkhe
Arizona State University
Email: dgurkhe@asu.edu
ID: 1209305002

Sriram Vellangallor Subramanian
Arizona State University
Email: svellang@asu.edu
ID: 1209270383

Karumpudi Ramakrishna Reddy
Arizona State University
Email: rkreddy@asu.edu
ID: 1209319770

***Abstract*—What we lack with conventional audio recording is meta-data about the recording telling us what is happening at different timestamps of recording. This project tries to solve this by adding meta-data to recordings by giving users the facility to put textual tags called hooks while they are recording. This allows the user to not just listen back to recording but also have time lined bookmarks of text to make him understand what was happened at different timestamps that he hooked a text.**

***Keywords*—Voice Recording, Playback, Waveform, Android, GPS**

## I. Introduction

Our application- the Intelligent Voice Recorder on Android platform- enables users to annotate the audio while recording, on the fly. This app will provide two functional modes- the recording mode and the playback mode. While recording an audio, the user can view the waveform of the recording real time. In the record mode, our user interface allows user to record an audio such as a class lecture and simultaneously insert hooks at desired intervals. These annotations can be input by keyboard. In the playback mode, user can view the waveform of the recorded audio as a time-line. Annotations would appear in timed sequence while the audio is being played. Finally, we provide basic file organization functions by which user can organize his notes based on location fetched from the GPS, tags, start time of recording or even his own preferences.

## II. Implementation

### A. Voice Recording

The voice recorder is built using the android audio capture library. The following bit of code is used to start recording and to save it later.

```
private void startRecording() {
    mRecorder = new MediaRecorder();
    mRecorder
    .setAudioSource(MediaRecorder
    .AudioSource.MIC);
    mRecorder
    .setOutputFormat(MediaRecorder
    .OutputFormat.THREE_GPP);
    mRecorder
    .setOutputFile(mFileName);
    mRecorder
    .setAudioEncoder(MediaRecorder
    .AudioEncoder.AMR_NB);

private void stopRecording() {
    mRecorder.stop();
    mRecorder.release();
    mRecorder = null;
}
```

To ensure the recording is not stopped when the app is pushed the background, all the recording activities are done in the background. The communication with the service and activity is done through MPI. The ability to hook text is added where user can add text while the recording is going on. Behind the scene important information like the location, and time is recorded. The locations is recorder at the start and at the end of the recording by using androids GPS location service[2]. The whole module is thoroughly tested, and also provides a custom build visualizer to show ripple effect based on amplitude while recording.

### B. Playback

Sriram

### C. Folder Organization

Rk

### D. Database and Storage

1) Database: We used sqlite database for storing information for each recording. The table had the following schema structure:
    a) created-at :DATETIME
    b) Filename :Text
    c) startTime :Real
    d) longitudeStart : Real
    e) latitudeStart : Real
    f) StartCity : Text
    g) longitudeEnd : Real
    h) latitudeEnd : Real
    i) EndCity : Text
    j) Time : float

    The data was required for various module like folder orgainization, UI adaptation, download.
2) Storage: For every recording apart from the above data which was stored in database, we also had to store the following data files in the storage.
    a) recording.3gp : The actual sound data which was recorded
    b) hooks.txt :Text file having all the timestamps for all the hooked text for the recording relative to the start time of the recording.

c) amplitudes.text : This was the time series of amplitude data collected during the recording. Which was used for better visualization during playback.

To bring an extra layer of security of the data we compressed all these file into a single file. This single file was given an arbitary extension of .drs, which won't let any user open the file using androids native decompressers.

## III. Status Table

TABLE I. Status table

| Serial No | Task | Assignee | Status |
|---|---|---|---|
| 1 | Voice recorder | Dhiraj | Completed |
| 2 | GPS tracker | Dhiraj | Completed |
| 3 | Note hooking | Dhiraj | Completed |
| 4 | Waveform timeline | Sriram | Completed |
| 5 | UI adaptation | Sriram (RK) | Completed |
| 6 | Playback mode | Sriram | Completed |
| 7 | Playback controls | Sriram | Completed |
| 8 | Folder organization I | RK | Completed |
| 9 | Folder organization II | RK | Completed |
| 10 | Folder organization III | RK | Completed |
| 11 | Download audio | Dhiraj | Completed |
| 12 | Visual spots | Sriram | Completed |
| 13 | Registration | RK | Completed |
| 14 | Background service | Dhiraj | Completed |
| 15 | Seeking to hooks | Sriram | Completed |

## IV. Screen shots

## V. Conclusion

## References

[1] http://developer.android.com/guide/topics/media/audio-capture.html

[2] http://developer.android.com/guide/topics/location/index.html