Dayton Gomez

CS442

Structure:

* /Source/
  + external/
    - User\_Interface class
      * I don't know alot about C# GUI's. This needs to be planned out by Jesse and Chase.
    - File class - handles creating and deleting of schedule files and events.
      * function addScheduleEvent(start, end, sound file path ) - returns an event ID.
      * function removeScheduleEvent(eventID) - true on success, false otherwise.
      * function modifyScheduleEvent(eventID, start, end, sound file path)
      * function saveFile(file path) - true on success, false otherwise
      * function deleteFile(file path)
    - Zip\_File class - extends file class?
      * function addFile(filePath)
      * function removeFile(filePath)
      * function saveZip(filePath)
  + internal/
    - A "main" file - This file will grab the event with a start time > NOW but < all other event's start time (the event coming up next). **This needs a good name.**
    - FileReader class
      * Construct(zip file path) - takes in the path to the zip that contains the sound files and XML schedule file.
      * private function parseXML(zip file path) -parses an XML file and returns some data type. in PHP this would probably be an associative array. Not sure about java.
      * function getEvent(eventID) - returns all data associated with an eventID.
    - Record class
      * Construct(save path) - if no save path is given, use the timestamp and save to default directory (var/recordings/)
      * function startRecord() - record to the given save path in construction
      * function endRecord()
    - Play class
      * Construct(sound file path)
      * function play() - plays the file given in construction
      * function stop()
    - Log class
      * write\_log(log level, log message) - generates a log message in the format: <Log Level>: <Timestamp> - <Message>\n to log file syslog.log within var/. Levels: INFO, WARNING, ERROR
  + var/ - holds the l
    - recordings/
  + etc/ - holds the zip. The zip holds the xml and sound files.
    - sounds/ - holds uncompressed sound files
    - events/ - holds the uncompressed xml file(s)

External Application (EA)(Need Name?)

GUI creates a zip file that contains any "imported" sound files as well as the "schedule.xml" file. This file needs to have a constant name (or directory) when it is placed within the internal application. The IA will then access that file whenever a scheduled event is activated or queued for activation.

Internal Application (IA)(Need Name?)

Process:

1. System starts
2. System loads our program.
3. The "main" file loads.
4. Main file finds the upcoming scheduled event.
   1. If the event is within some quick threshold, leave system active and wait for play / record
   2. If the event is past the threshold, suspend system until **X** seconds before event occurs.
   3. **IF NO EVENT EXISTS: Shut down?**
5. If play data is given within the event AND the play length is > 0 AND a sound file is given:
   1. Play sound file
6. If sound file has played, stop the play using the given length.
7. If record data is given within the event AND the record length is > 0:
   1. Start Record
8. If recording began:
   1. stop recording. Save file to disk. Zip the recording. Make sure name of the sound file is related to its start/stop record timestamp.
9. Go back to step 4.

Repeat steps 4 - 9 forever. The box will either run out of events or have its battery die.

Main File

Record

FileReader

Play

Log