Incremental && Regression Testing

Team 15 - *Beefed Up Music Scheduler*

Team members

William Vanschaik (wvanscha@purdue.edu)
Joey Imburgia (jimburgi@purdue.edu)
Santiago Abondano (sabonda@purdue.edu)
Rachel Gully (rgully@purdue.edu)
Gaurav Srivastava (srivast6@purdue.edu)

Introduction

During this sprint, our chosen form of incremental testing was **bottom up**. We started by unit testing the individual classes, building on top of the individual classes, and then testing the combinations until reaching the highest level (GUI).

Classification of Components

User Interface (UI)
This is the User Interface, which will allow the user to control the functionality of our program with a graphical interface. The UI includes the following items: Music Controls, Playlist Selection, Volume Slider, and Scheduling Controls. This class will allow a user to select a song or a playlist, which are the inputs. The song or playlist selection are saved as global variables to be referenced later in the audio playback and playlist modules.

Music Controls
The music controls allow the user to play, pause, or stop the current song. These functions depend on the Audio Playback module. When a user presses any of the buttons, that action is passed to the Audio Playback module.

Playlist Selection
This allows a specific playlist to be selected and to view the songs inside a playlist. The input is a mouse click to choose a specific playlist to view or play.

Volume Slider
The volume slider allows a user to control the gain of a song. The input is the location that the slider is currently at on the slide bar.

Scheduling Controls
These controls allow a user to schedule a song or a playlist to play at a specified time. The input is a button click or text input of a specific time.

Audio Playback
This class focuses on opening an audio file and controlling the flow of playing, pausing, and stopping a file. The input is an mp3 file located in the subdirectory of one of the playlist folders.

Playlist Controls
This will control how a group of songs is played. The input is action clicks on the buttons. The output can be a new playlist.

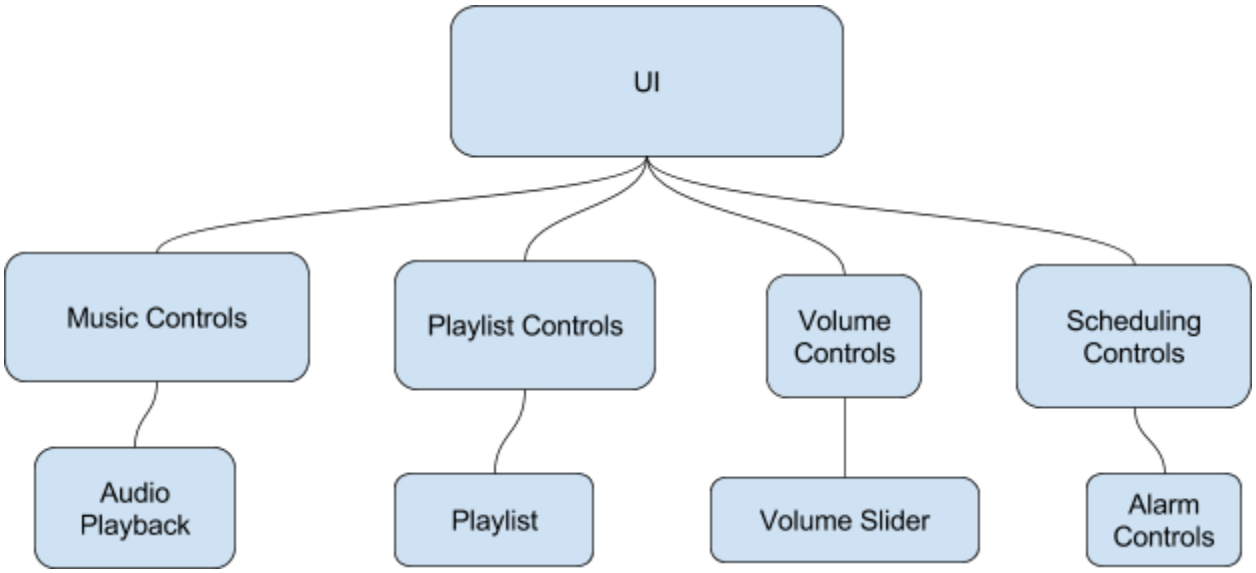
Playlist
This will allow a user to group certain songs together to be played in sequence. The input is the folder containing the playlist.

Volume Controls
This will control the volume of the output. The input is the levels from the volume slider.

Scheduling Controls
Will create an event to play a song at a specified time. The input is the user selections of time and object to play. The output is a scheduledPlay instance.

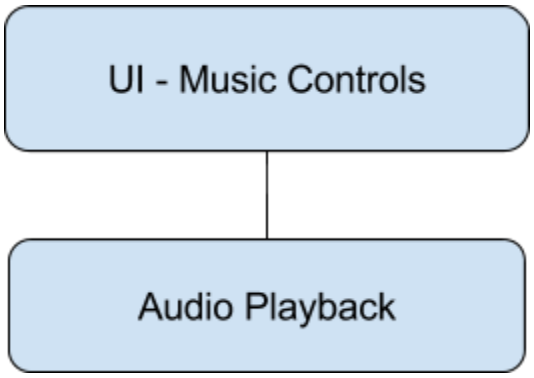
Alarm Controls
This will play a song as an “alarm”. This is triggered by a scheduled event. The input is the user selected time and the output is a scheduled alarm.

Class Hierarchy Diagram



Regression Testing Log

Severity 1
Severity 2
Severity 3



Module	UI-Music Controls

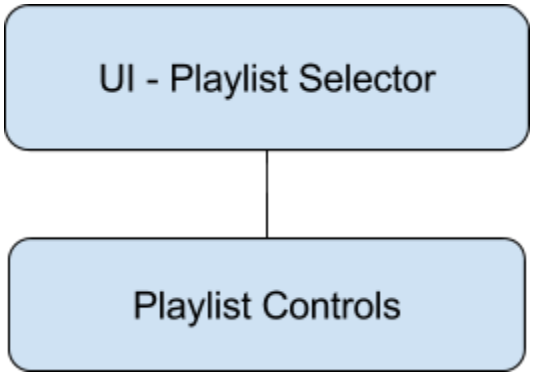
Incremental Testing

Defect #	Description	Severity	How To Correct
1	Pause and resume buttons should not be visible when a song is not playing.	1	Only make pause and play buttons visible after the song has started to play.
2	After song playback is finished, the application does not get notified when the play thread is ended and song has stopped.	1	Create a new thread to watch the pool of music threads and notify the UI when playback has completed.
3	Playing a song without selecting a directory gives an error.	1	Displayed an alert dialog to make sure a user has selected directory before playing a song
4	There is no next button, and you cannot change songs mid playing songs.	2	Added a next button and created a function to change the song to the song you selected.

5	Song title not currently updating based on what song is actually playing.	3	Added a label and whenever a new song is starting to play the label will be updated to the current song.
6	Music Playlist directory does not persist when application is closed.	3	Added a serialization of the directory location and loads it on startup.

Regression Testing

Defect #	Description	Severity	How To Correct
1	Fixing pause and resume buttons and making them visible only after starting playback causes the UI to display the buttons still after the playback has stopped.	1	Check for completion of song or button click for stop and set the visibility to false.
2	While checking for song completion with a thread, the UI does not get updated as soon as the song is complete.	1	When complete notify the UI to update the buttons.
3	Song title showing a default song title when program first starts	2	Make the JLabel invisible when no song is playing or has stopped
4	The serialization did not load on startupon properly as just a line of text.	3	Created a startup object which contained several preferences that was loaded on startup to keep object integrity.



Module	Playlist Usage

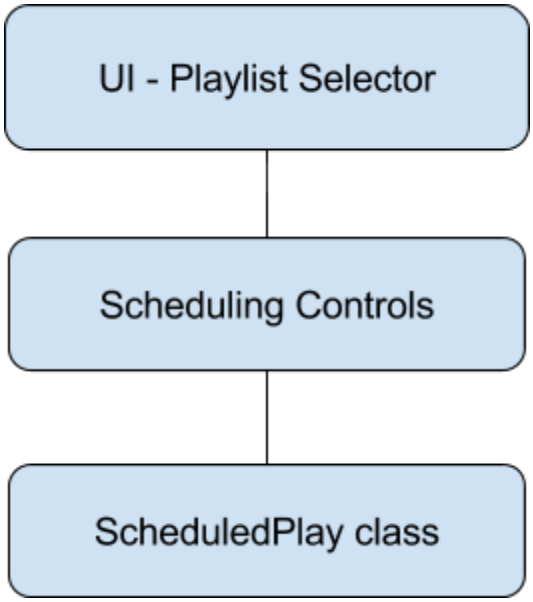
Incremental Testing

Defect #	Description	Severity	How To Correct
1	Playback functionality not fully integrated with Playlist class	1	add function call to play in the noted location
2	Folder system is not currently implemented, and playlists cannot be created from the UI.	1	Create a function and UI tab to allow creation of playlists using the new folder system.
3	When selecting directory, user is unable to choose a directory from another location	1	Changing the directory gui making it possible to go to different directories
4	Unable to list songlist in a playlist directory	1	fixed the file.list to only display mp3 files
5	Playlist does not have all needed fields	2	Added all needed fields
6	Unable to find playlists inside a directory	2	fixed the file.list to filter only subdirectories that represent playlist
7	Playlist does not shuffle songs	3	Create a random number generated and give each song an index and select a song from the playlist folder and add them into the queue.
8	Playlist repeats songs when shuffling	3	Remove a song from an arraylist whenever the song is played and use the arraylist to queue songs.

Regression Testing

Defect #	Description	Severity	How To Correct

1	Playlist not advancing to the next song when current song ends	1	advance to next song in the playlist when thread notifies
2	Shuffle algorithm uses the same random each time, thus not really shuffling the songs.	3	seed the random before reuse.
3	Playlist save function does not save name correctly with new playlist folder system	3	correct save function to change folder name



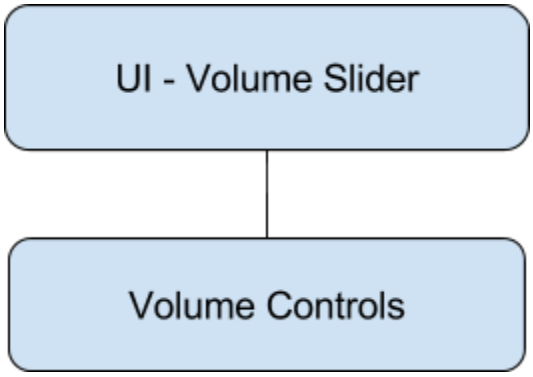
Module	Scheduling Song or Playlist

Incremental Testing

Defect #	Description	Severity	How To Correct
1	scheduled events do not persist after the application is closed.	1	Use save function to writed scheduled events to file and then add them on startup

Regression Testing

Defect #	Description	Severity	How To Correct
1	Scheduled song or playlist does not run correctly if there is already an active song or playlist	2	add a check to the scheduled event to pause what is currently playing
2	Scheduled events not written to file if application closes unexpectedly	3	write scheduled events to file immediately and remove file when event is triggered



Module	Volume Controls

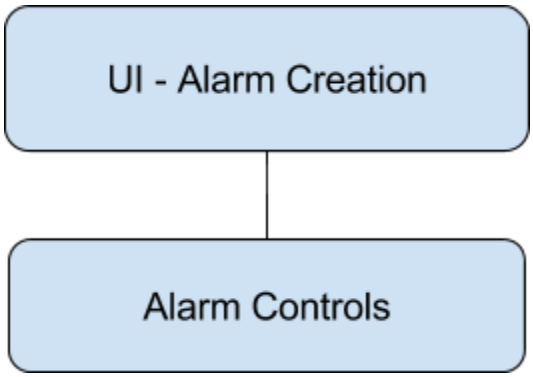
Incremental Testing

Defect #	Description	Severity	How To Correct
1	Volume should update correctly based on the value of the volume slider.	1	Create a change listener to update the volume when the slider value changes.
2	On start of new song the volume was set back to its default value, and not persisting to the slider's location.	3	Created a global variable for volume and made it not song specific.

Regression Testing

Defect #	Description	Severity	How To Correct

1	Adding the change listener didn't correctly update the volume.	2	Fix the formula used to update the volume, so it is within the correct range.



Module	Alarm Control

Incremental Testing

Defect #	Description	Severity	How To Correct
1	Alarm not scheduling for future date	1	Convert the future date to seconds and schedule task seconds in advance
2	Alarm snooze not working when alarm does go off	1	Add snooze boolean to the Alarm class, along with the appropriate helper functions.
3	Alarm snooze currently has a predetermined length for snooze	2	Add a snooze_length variable in the Alarm class, along with the appropriate helper functions.
4	Multiple alarms can go off simultaneously.	3	If an alarm is going off there is a boolean that lets the application know there is an alarm going off so another alarm may not go off.

Regression Testing

Defect #	Description	Severity	How To Correct
1	No way of telling which alarms are on and scheduled	1	Create a list of the alarms currently scheduled
2	No way of tellings which alarms have been created but not schedule	2	Create a list of the alarms created but not scheduled
3	There should be an option to tell the Alarm is recurring	2	Add snooze boolean to the Alarm class, along with the appropriate helper functions.