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.

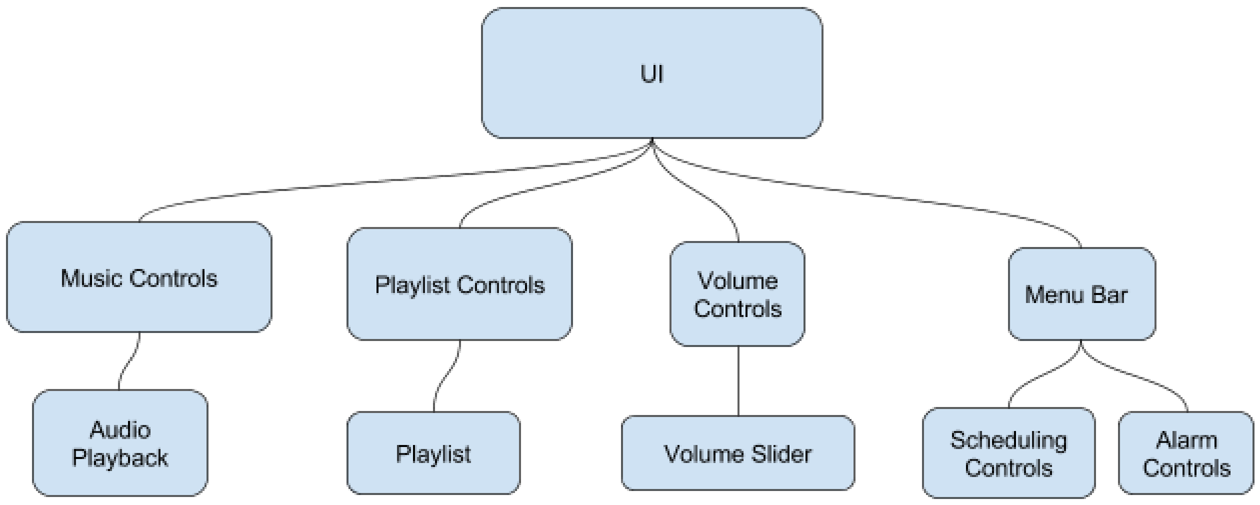
Scheduling Controls
Will create an event to play a song at a specified time. The input is the user selections of time and object to play. The output is a scheduledPlay instance.

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.

Song Queue
This will take in a song to play after the current song is over. The input is a user pressing a specific button to add the song into the queue. The queue will show the order of the songs to be played next.

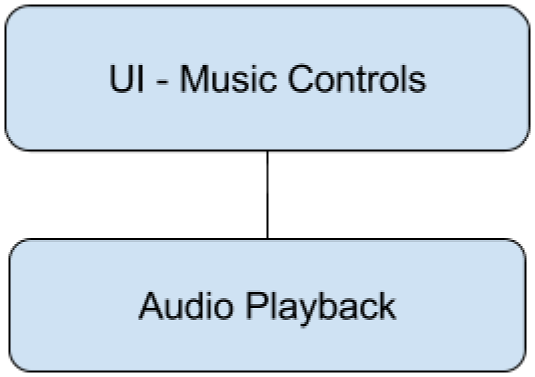
Menu Bar
This will be the way to manipulate the schedules and alarms. Creation, editing, and deleting of alarms and schedules will be done through here. It also allows the program to change directories.

Class Hierarchy Diagram



Regression Testing Log

Severity 1
Severity 2
Severity 3



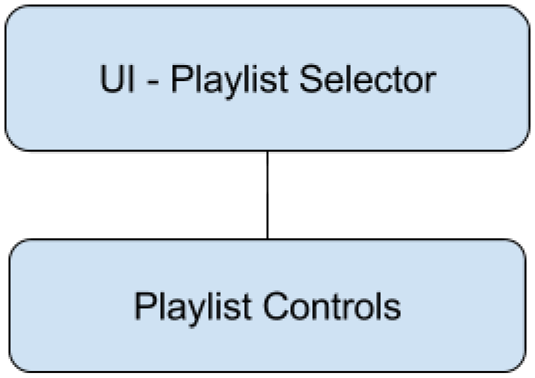
Module	UI-Music Controls

Incremental Testing

Defect #	Description	Severity	How To Correct
1	When changing the directory a new main panel is created, this keeps the existing mainPanel and displays the new one underneath the old one.	1	Delete the old mainPanel, and recreate with with the new playlist directory mainPanel.
2	When selecting a song to play, if a song is already playing or paused the song would resume at that location in a different song.	1	Do not allow the songs to be changed while another song is playing or paused until the song is stopped.
3	If you go to different playlists you can get multiple songs to play at once if you stop and resume them at different intervals.	2	Keep a boolean value that keeps track to see if a song is playing and do not allow the start of a new song in another playlist if a song is currently playing.

Regression Testing

Defect #	Description	Severity	How To Correct
1	When keeping the existing jFrame, when the directory is changed, the application freezes and does not respond to any input and must be restart.	1	Delete the components inside the jFrame and recreate the components inside.



Module	Playlist Usage

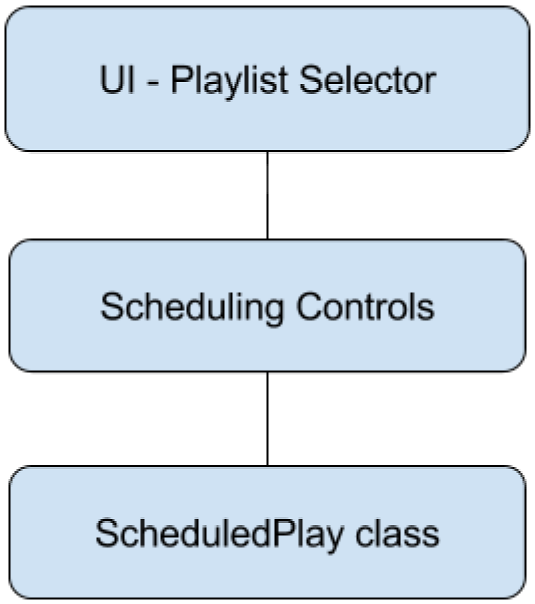
Incremental Testing

Defect #	Description	Severity	How To Correct
1	Selecting a directory that does not contain a playlist with songs gives error	1	Instead of giving an error it should not display any playlists.
2	If there are no folders in your playlist directory, there will be a null pointer exception on startup of the application.	1	Create empty directory folders to show that there is nothing in that directory instead of passing a null value.
3	Double clicking a song while another song is playing plays both songs simultaneously	2	Stop the song that is currently playing and play the new song that is double clicked

4	Double clicking a playlist does not play the songs in it	2	Create a queue that includes all the songs in the queue and set the music player to the beginning of the queue
6	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	fix issues from recent changes to get it working again
2	Shuffle algorithm uses the same random each time, thus not really shuffling the songs.	2	Change shuffle implementation in playlist class to address this issue.
3	No way to change playlist name after playlist is created	3	add a function in the playlist class to change folder name and put an option in the gui if needed



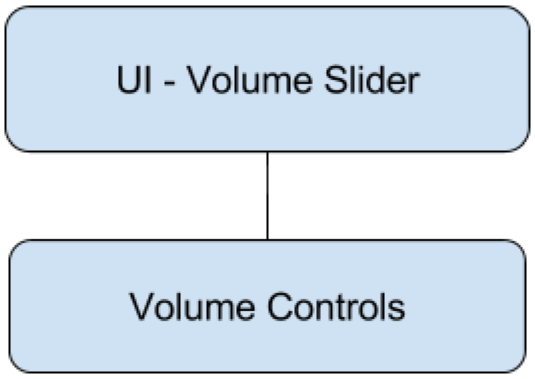
Module	Scheduling Song or Playlist

Incremental Testing

Defect #	Description	Severity	How To Correct
1	scheduled events are saved in a text file instead of the new savedata class	1	Change the savedata class to incorporate scheduled plays and fix the save function in the scheduledplay class
2	scheduled events require a playlist to multiple songs	3	add functionality to automatically create a playlist for scheduling multiple songs
3	scheduled events do not take into account daylight savings time	3	check to see if system time has changed when application loads scheduled events

Regression Testing

Defect #	Description	Severity	How To Correct
1	Scheduled events not written to new savedata if application closes unexpectedly	2	when updating to the new save class add functionality to save event automatically on creation to prevent loss



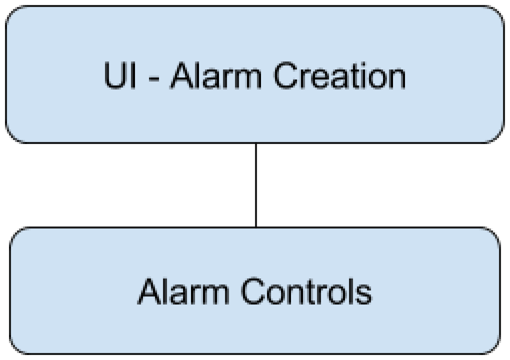
Module	Volume Controls

Incremental Testing

Defect #	Description	Severity	How To Correct
1	Volume won't change if the program is run on a different platform (OSX vs Windows) than the development platform.	2	Make sure to add in case statements so that the correct Master Volume line is chosen; search for multiple line names instead of just one.

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 Snooze not rescheduling after hitting snooze	1	Schedule a new task alarm with snooze length as a future date
2	When rescheduling new alarm date is created with wrong year	1	Force current year on object creation
3	When storing the future date in seconds doesn't the conversion can result in a long	2	Store seconds in a long
4	Alarms recurring go off instantly	3	Make sure when rescheduling a recurring alarm it's for the next date

Regression Testing

Defect #	Description	Severity	How To Correct
1	Depreciation with the date object could cause problems	1	Switch to the calendar to keep track of date
2	Managing single alarms and snoozing rescheduling and recurring alarms is getting confusing	2	Split them up in 3 different classes