Today I'm going over an MP3 player interface that I helped design in college. I worked on this project with 2 other individuals (shout out to Wilburn and Michael). We created our own Graphical User Interface (GUI) for our MP3 player and used the mp3agic library to read mp3 files and ID3 tags. We incorporated a database into our design to help us store song information and display it within our GUI.

I was in charge of the database aspect of our design. This involved making sure the appropriate Create Read Update Delete (CRUD) operations were present in the design. This means retrieving the appropriate data when buttons are pressed on the GUI, deleting data from tables, adding data to tables, and modifying data when appropriate. Once I finished the core CRUD operations within the database, I then helped put the finishing touches on the GUI and I also helped connect the different aspects of the project together so that they functioned properly and cohesively. I even revisited this project recently and added in a few additional bonus features, which I will talk about later.

Our MP3 Player can successfully play, pause, and stop songs. This functionality is available to both songs inside of the user's library and MP3 files that the user wants to open from a different location. These three songs are already in my library. Let me demonstrate this. Now let me open up a song not in my library. We also give users the options to skip to the next song or skip to the previous song using the appropriate buttons or the control menu. We also implemented keyboard shortcuts to help the user quickly play and skip through songs. To quickly play a song, all the user has to do is select the song they want to play and press the space bar on their keyboard. To skip to the next song, the user just needs to press ctrl and the forward arrow on their keyboard. To skip back to the previous song, the user just needs to press ctrl and the backward arrow on their keyboard. Users can also add songs to their library through the file menu or by simply dragging and dropping the desired mp3 files into their library. Deleting songs is just as easy. Users can either select the songs they want deleted and delete them through the file menu or they can simply right-click the songs and choose the delete option.

Playlists are also implemented into our MP3 player where users can easily create and delete playlists as well as add songs to and delete songs from playlists. When the user selects a playlist name from the side tree, the window displays all of the songs currently in that playlist. Users can also conveniently increase/decrease the volume of the songs being played by using the volume slider in the upper left of the GUI or by using the Control menu. They can also use the keyboard shortcuts of crtl + I for increasing volume and ctl + d for decreasing volume. Users can also choose to repeat songs after they finish and shuffle the songs in their library or playlists by clicking the appropriate options in the Control menu. These features take effect once the song finishes playing.

The song information area is also user-customizable. They can choose what song information they want to be visible by right-clicking the information header. They can choose to displaying the artist, album name, release year, genre, and comments of songs in their library.

Finally, users can track the progress of the song being played by viewing the song progress bar at the very bottom of the GUI. This progress bar displays the current time elapsed, and the total time remaining until the song is over.

I added in the additional functionality of playing the next track right after the current track finishes playing (when shuffle or repeat are not selected of course). Before we did nothing after the current track finished playing. Now it automatically starts playing the next track. I also added in the functionality of allowing the user to skip anywhere in the song by clicking on the desired spot inside of the progress bar. Before the user was only able to view the progress of the song. Now they can use the progress bar to skip to a particular part in the song.

Now that I've talked about what the MP3 Player can do, let me skim through the code so that I can show you how these different features are achieved. Here is the actual database that is being used in our project. As you can see it's called MP3\_Player\_Database. We have a couple of tables that we use throughout the MP3 Player. We use this table to determine what columns the user wants to display in their library. We use this table to hold all of the information of the songs that are in our library. We also hold the path value of where the file is located so that the user can add songs to their library from any location on their computer without issues. This table is used to hold the names of the playlists that the user has created. This is just a stubborn duplicate table that really doesn't want to be deleted. Whenever I try to delete it, it just closes the connection to the MP3\_Player\_Database and I have to re-establish it again. This is a very simple table that keeps track of which songs are associated with which playlists. Instead of redundantly keeping all of the song and playlist info, I simply store the keys for the desired song and playlist and then use this information to display the correct songs whenever the user wants to view a particular playlist. And finally, this table is used to store the 10 most recently played songs so that the user can view what songs they have recently played.

Here is my Library class. It's a simple wrapper class used to provide a level of abstraction from all of the queries going on in the internal MP3\_Database class. One important thing to mention is that I have to make sure I'm actually connected to the database using the services tab before I run the program. Otherwise it will obviously fail to connect to the database and it won't be able to display any of its data. Anyways, here is where the MP3\_Database class starts. Here is the code used to create the various tables. Here is the code used to delete the tables. Here is the code used to add rows to the different tables. Here is where I start deleting rows from the tables. And finally here is the code used to retrieve the appropriate data from the tables.

Lets move on to the Main\_GUI class. Here are all the different variables needed throughout the program. Here is the timer variable that I use to update the progress of the song being played. Here are all the different buttons used. Here is the slider that is used to control the volume of the music being played. Here are the different menu items. Here is the table header that the user can interact with and select what information they want to be displayed. Here is the tree used to display the different library and playlist names. Here is the Mouse-Listener used to determine what data to display based on what item in the tree the user selects. Here is the progress bar information and the Mouse-Listener used to skip to a specific part in the song if the user clicked somewhere on the progress bar. Here is the FileDrop implementation that allows the user to drop files into the library to automatically add them to it. Here are the different keyboard shortcuts implemented in the program. Here are the different event handlers used. Here is the method used to update the progress bar information. This method is used to display the time in a hour, minute, second format.

Here is what happens when a forward action is performed. We skip to the next track or loop back to the beginning if the current track is the final track in the library or playlist. Here is what happens when a play action is performed. We either play the song that the user opened or we play the selected song from the library or playlist. While doing this we keep in mind if we are resuming a song that was previously paused, playing a song from the beginning, or skipping to a particular point in a song and playing from there. Here is the pause action handler code. We preserve the state of the song that we just paused and stop the timer. Here is what happens when a stop action is performed. We basically reset all the appropriate variables and close the player. Here is the backwards action handler. We skip to the previous track or loop to the end if the current track is the first track in the library or playlist. Here is where we are adding all of the menu items to our file menu and control menu. Here is where we are selecting the song that the user clicks on.

These are the handlers for increasing and decreasing the volume. Here is the handler for adding a song to the library and here is the handler for deleting a song from the library. This is the code that is run whenever the user opens an MP3 file. Here is where we add playlist songs and here is where we delete playlist songs. Here is the code used to create a new playlist. Here is where we populate the recently played songs list. Here are the handlers for the replay action and the shuffle action. Here is the code that modifies the different columns that the user wants to display. Here is where we delete songs from all playlists that they were in if they were removed from the library. Here is where we are update what songs are being shown to the user. And finally, this last bit of code is where we initialize the GUI and show it to the user. That about wraps up this demo. As always, thank you for watching and have a great day.