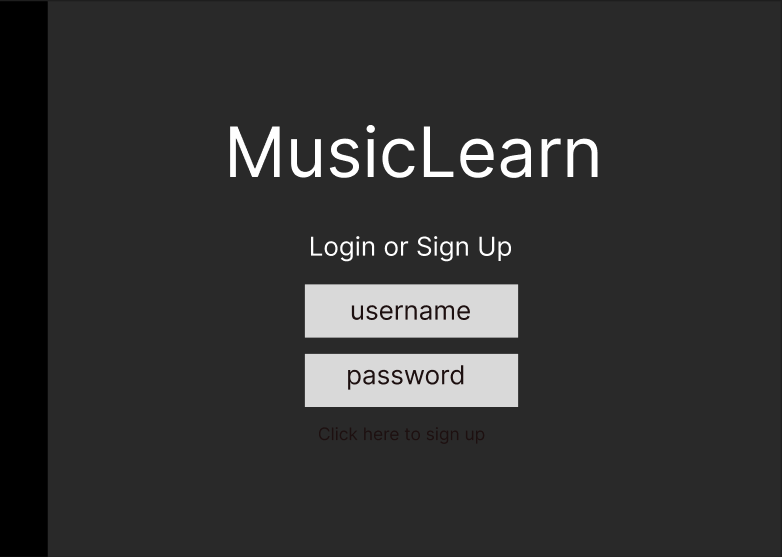
Design-2

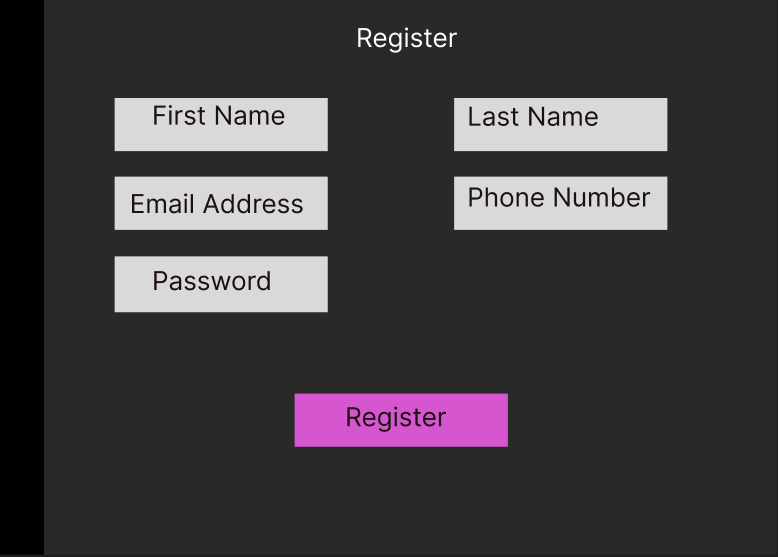
**UI DIAGRAMS**

**Figma board link:** [**https://www.figma.com/file/ZWFEKygcQvkxaPiJxfqcFd/songteacher?type=design&node-id=0%3A1&mode=design&t=LjmFiflXxgftSRw9-1**](https://www.figma.com/file/ZWFEKygcQvkxaPiJxfqcFd/songteacher?type=design&node-id=0%3A1&mode=design&t=LjmFiflXxgftSRw9-1)

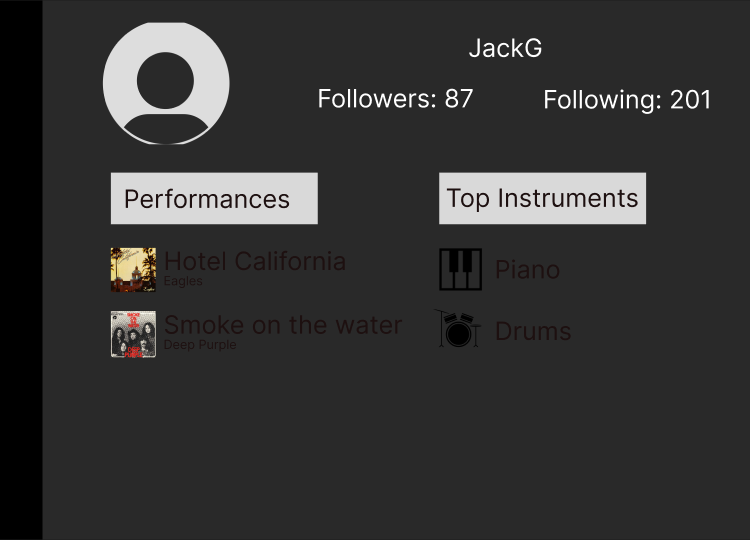
**Login**

****

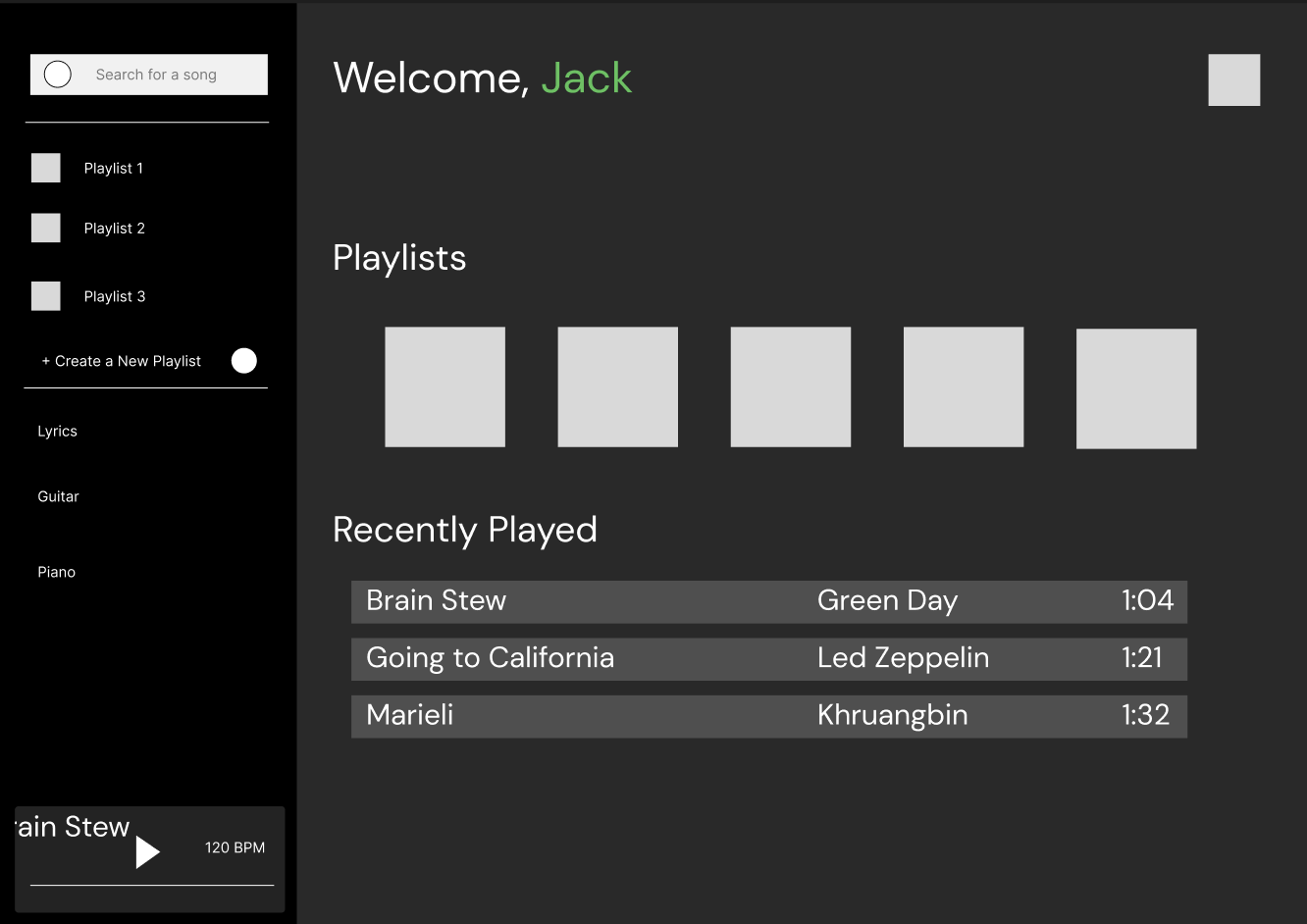
**Register**

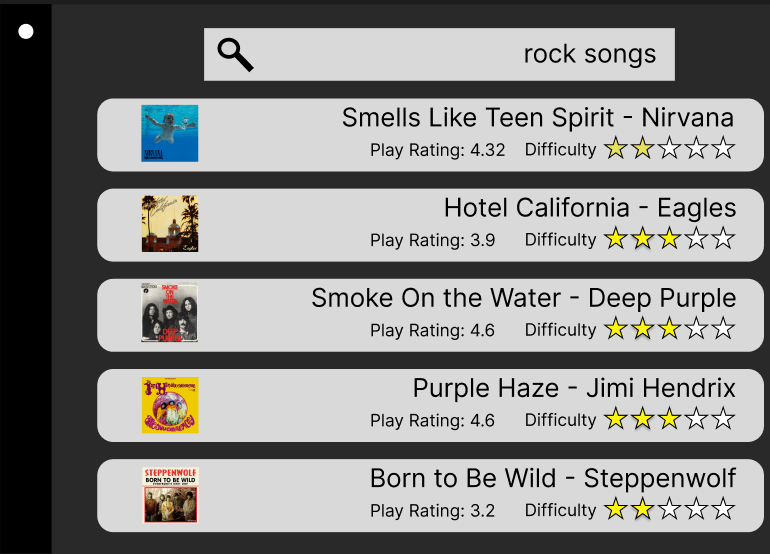
****

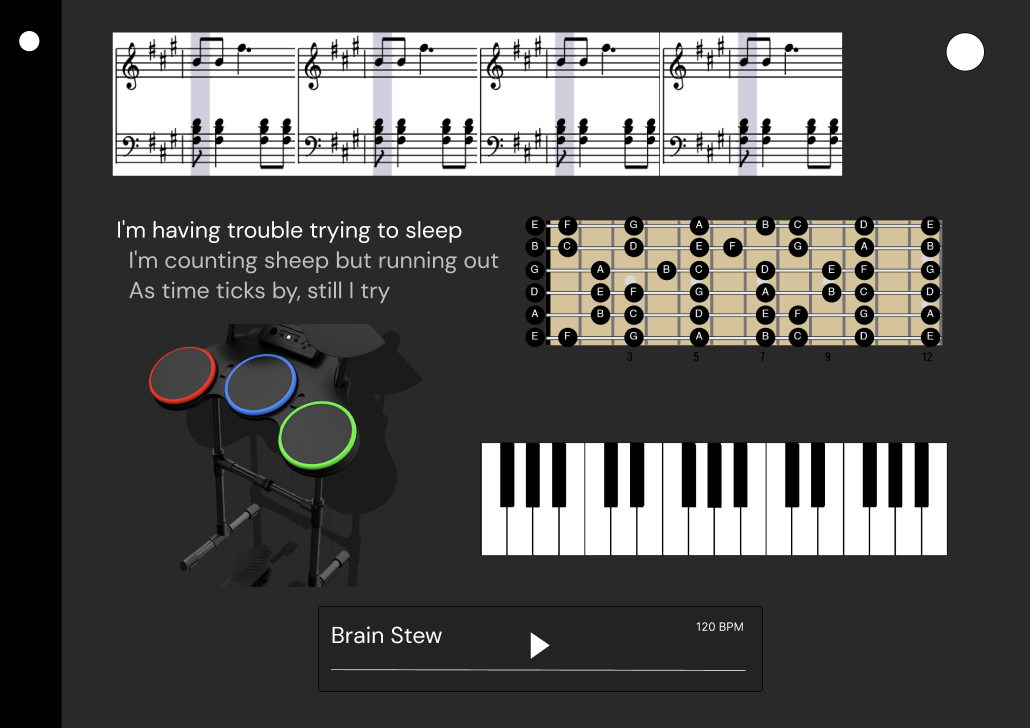
**Profile**

****

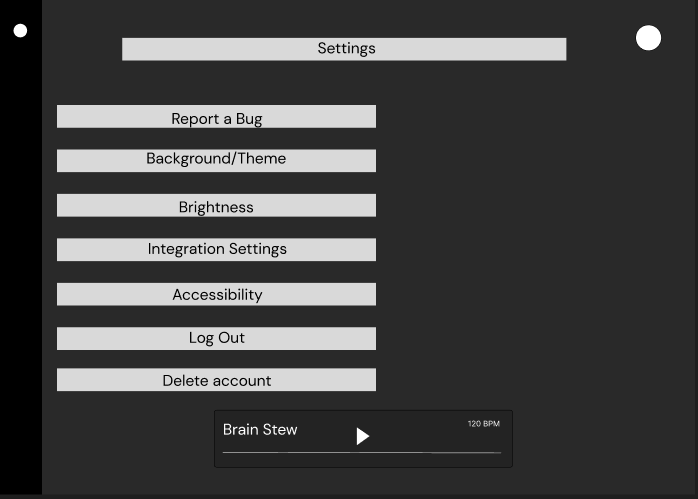
**Main View (after login)**

****

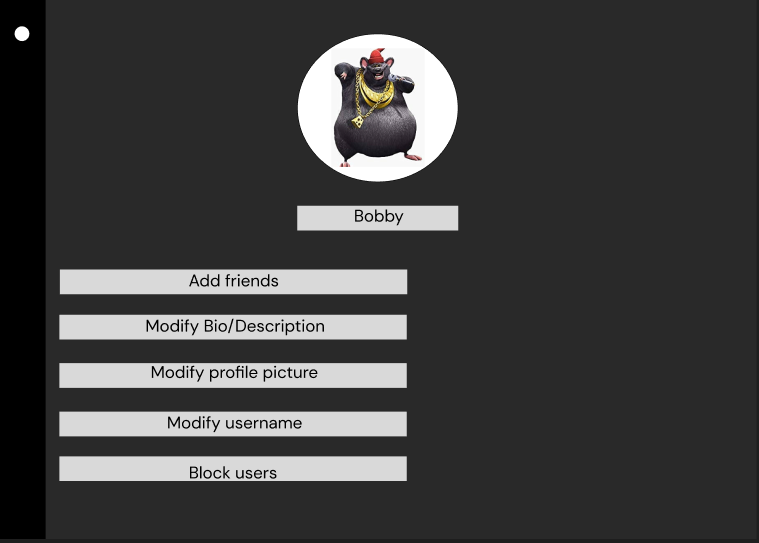
**Search for Song View**

**Play Song View  
**

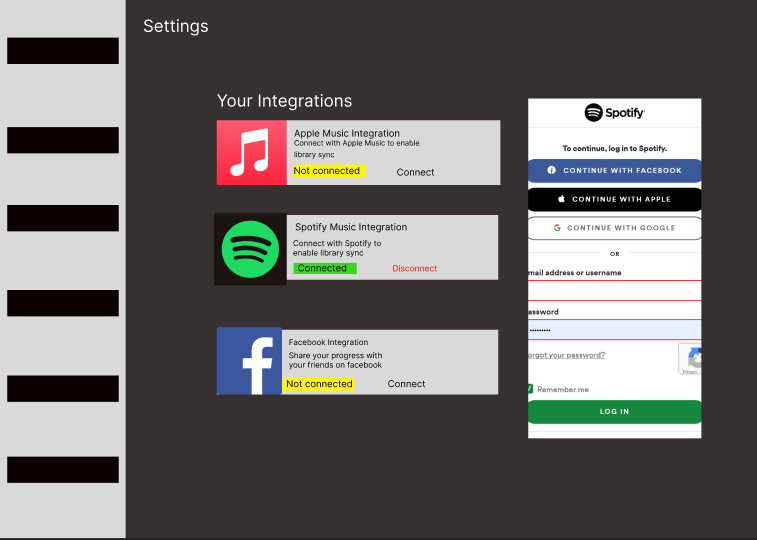
**Settings**

****

**User Settings**

****

**External App Connection Settings**

****

**PSEUDOCODE  
// Search for a Song**searchForSong(search: String, availableSongs: List)

FOR int i from 0 to the length of availableSongs:

IF name of song at index i equals search:

return song at index i

ENDIF

ENDFOR

return no\_song\_found

**// Play a Song**

playSong(song: Song)

IF song is not null:

start playing song

ELSE

display error message "Song not found"

ENDIF

END

**// Create a Playlist**

createPlaylist(name: String, songs: List)

playlist = new Playlist

playlist.name = name

playlist.songs = songs

return playlist

END

**// Add Song to Playlist**

addSongToPlaylist(song: Song, playlist: Playlist)

IF song is not null AND playlist is not null:

add song to playlist.songs

ELSE

display error message "Cannot add song to playlist"

ENDIF

END

**// Rate a Song**

rateSong(song: Song, rating: Integer)

IF song is not null AND rating is between 1 and 5:

song.rating = rating

ELSE

display error message "Invalid rating"

ENDIF

END

**// Shuffle Playlist**

shufflePlaylist(playlist: Playlist)

IF playlist is not null AND playlist.songs is not empty:

randomize playlist.songs

ELSE

display error message "Playlist is empty"

ENDIF

END

**// Display User Profile**

FUNCTION displayUserProfile(userId):

user = getUserById(userId)

IF user is not null:

PRINT "User Profile Picture:"

PRINT user.username

PRINT "Followers: " + followers

PRINT "Following: " + following

PRINT "Total Songs: " + length(user.songs)

IF length(user.songs) > 0:

PRINT "Performances:"

FOR i FROM 0 TO length(user.songs) - 1:

song = user.songs[i]

PRINT " " + (i+1) + ". " + song.title + " - " + song.artist

ENDFOR

ELSE:

PRINT "No Performances."

IF length(user.instruments) > 0:

PRINT "Top Instruments:"

FOR i FROM 0 TO length(user.instruments) - 1:

song = user.instruments[i]

PRINT instrumentIcon[i] + “ “ + user.instruments[i]

ENDFOR

ELSE:

PRINT "No Performances."

ELSE:

PRINT "User not found."

END FUNCTION

**// Change Settings**

function changeProfileSettings(user, newSettings):

if user is\_authenticated():

// Assuming user is logged in

// Fetch existing profile settings

currentSettings = user.getProfileSettings()

// Merge new settings with current settings

updatedSettings = merge(currentSettings, newSettings)

// Validate the updated settings

if validateSettings(updatedSettings):

// Save the updated settings to the user's profile

user.updateProfileSettings(updatedSettings)

// Inform the user that settings have been updated successfully

displayMessage("Profile settings updated successfully.")

else:

// Inform the user about invalid settings

displayMessage("Invalid profile settings. Please check and try again.")

else:

// User is not authenticated; prompt them to log in

displayMessage("Please log in to change profile settings.")

// Helper function to merge current settings with new settings

function merge(currentSettings, newSettings):

mergedSettings = {}

// Copy current settings

for setting in currentSettings:

mergedSettings[setting] = currentSettings[setting]

// Update with new settings

for setting in newSettings:

mergedSettings[setting] = newSettings[setting]

return mergedSettings

// Helper function to validate the updated settings

function validateSettings(settings):

// Implement validation logic based on your application's requirements

// For example, check if email address is valid, username is unique, etc.

// Return true if settings are valid, false otherwise

return true

// Helper function to display messages to the user

function displayMessage(message):

// Implement a function to display messages in your application's UI

// For example, show a notification, alert, or toast message

print(message)

**// Grade Piano Tempo**

FUNCTION calculateTempo(playedNotes: List, expectedNotes: List, tempo: int) -> int:

Let StartScore = 100

IF length of playedNotes is not equal to length of expectedNotes

StartScore - -

ENDIF

RETURN StartScore

END FUNCTION

**// Grade Piano Accuracy**

FUNCTION calculateAccuracy(playedNotes: List, expectedNotes: List) -> int:

LET correctNotes = 0

FOR int i from 0 to length of playedNotes:

IF playedNotes[i] == expectedNotes[i]

correctNotes++

ENDIF

ENDFOR

LET accuracy = correctNotes / length of playedNotes

RETURN accuracy

END FUNCTION

**// Performance Score**

FUNCTION performanceScore(playedNotes: List, expectedNotes: List, tempo: int ) -> int:

Let accuracy = calculateAccuracy(playedNotes, expectedNotes)

Let tempo = calculateTempo((playedNotes, expectedNotes, tempo)

RETURN (accuracy \* tempo)

END FUNCTION