**ITCS 4155**

**SD2: Software Design Document**

**1. Project Overview**

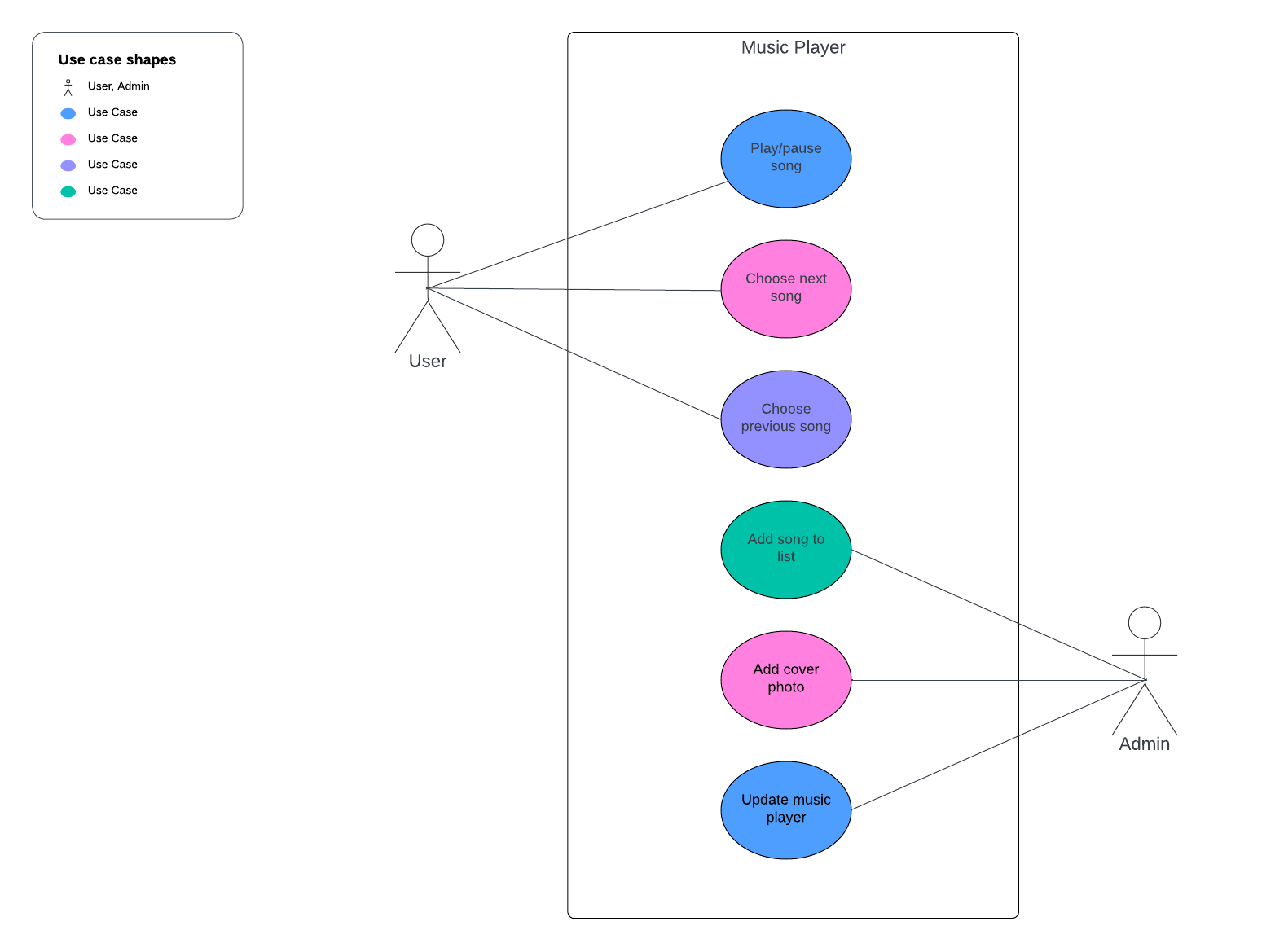
The purpose of the music player is to allow users to listen to and have access to music even when there is no internet connection. Most music apps, such as Spotify and YouTube music, require access to the internet, but this music player can play songs offline. It is useful in situations where users don’t have internet access, such as during air travel. Users can play and pause songs, skip songs, and go back to the previous song with the click of a button. Admins can change the layout of the music player, add or remove songs, and change album covers. Users can also become admins themselves by changing the codebase.

**2. Architectural Overview**

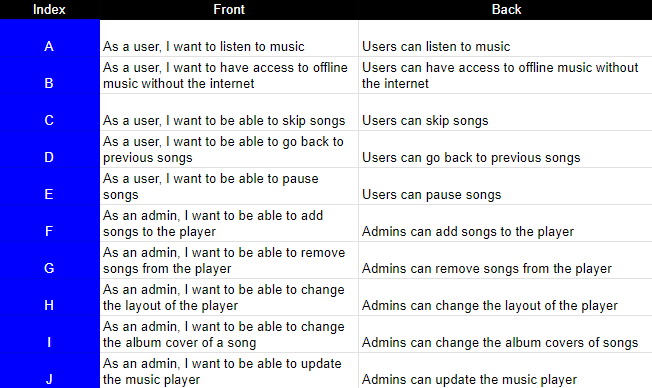
The music player was built using only HTML, CSS, and JavaScript. While Python was considered, I decided against using a backend since none was required for this project. Instead, the music player operates entirely on the client-side. To use the music player, the user will need to simply clone the repository on their device and open the **index.html** file on their browser.

This music player is a good alternative to Spotify and YouTube music because it enables offline listening, although Spotify has a feature that allows premium users to save songs to listen offline. However, unlike the other music services, this music player is completely free. Users can listen to their saved songs without the need for a premium service.

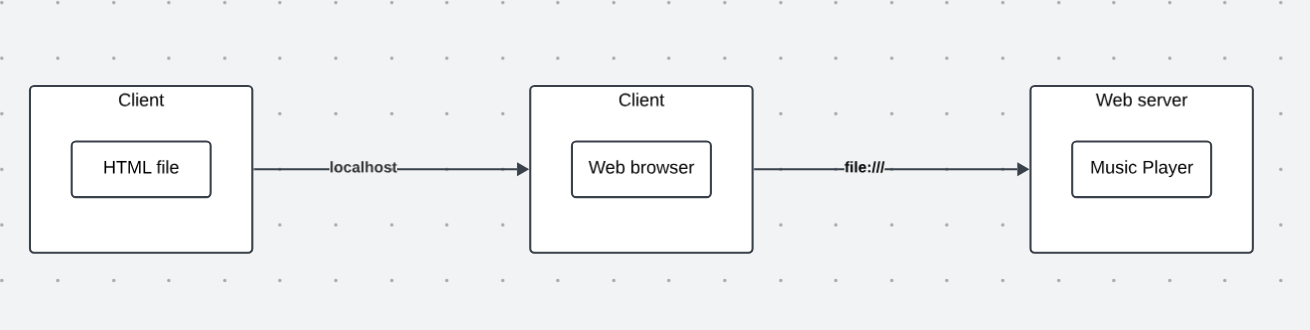
**Use Case**

****

**User Stories**

****

**Deployment Architecture**

****

Clients can connect to the music player by clicking on the HTML file, which automatically opens in their default web browser without needing internet connection. Since the music player is built using frontend languages (HTML, CSS, JavaScript), it can be easily deployed on a variety of devices. The simplest method is to deploy it on GitHub pages, which offers free hosting for static websites. Users can clone the repository and view it on their browser. There is no need for a backend server or database, which makes deployment lightweight and simple.

* Option 1: GitHub Pages – Easy and free option for public deployment
* Option 2: Local Deployment – Users can download the project and run the player locally on their browser

**Persistent Data Storage**

As there is no backend, the music player does not have persistent data storage. However, it does use HTML local storage. When the page refreshes, the music player automatically goes to the beginning of the first song in the list as that is the default starting point. This default position is set as the local storage location. This limited amount of storage requirements eliminates the need for a server-side database and ensures a completely offline experience.

**Global Control Flow**

The music player is procedure-driven, as it is a linear process where the user follows the same procedure upon using the system. The different controls and functionalities consist of the play and pause buttons, the song selection buttons, the shuffle button, and the background image for each song.

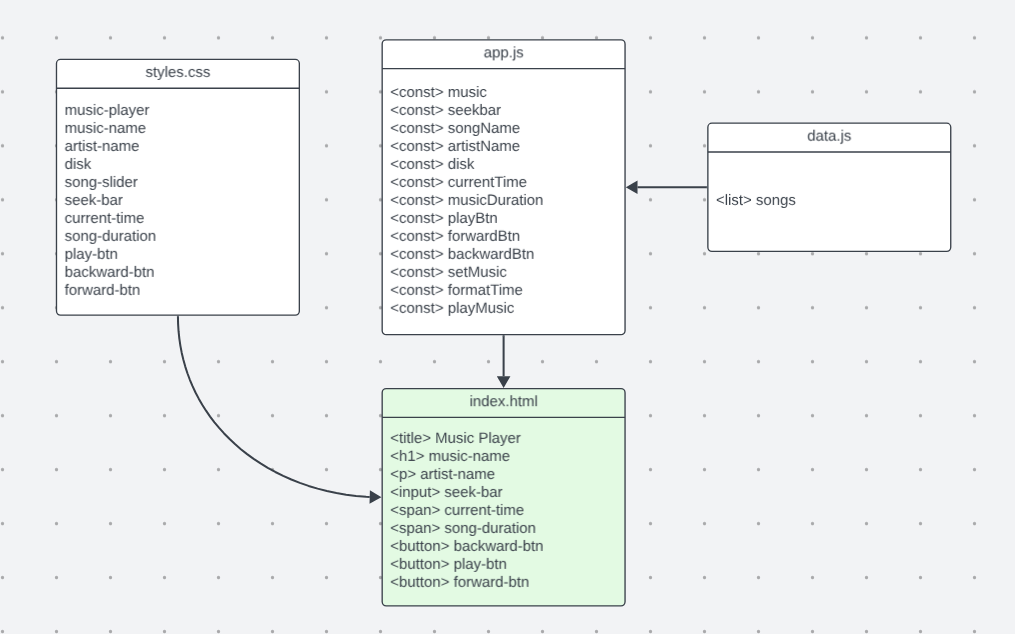
* **Main controls**: Play, pause, next, previous, and shuffle buttons are all handled via JavaScript event listeners
* **Seekbar**: Users can use the bar to select any point in the song, and the song duration will update automatically
* **Admin functions**: Admins can modify the songs files and song covers by editing the data.js JavaScript file directly. They can add song files to the music folder and add song covers to the images folder. With those new files, they can add them to the songs list in data.js.

**3. Detailed System Design**

The music player was built using the frontend programming languages, HTML, CSS, and JavaScript. The files included are index.html, style.css, data.js, app.js, and the images and music folders. The index.html and style.css files control the layout of the music player, such as the background colors, disk size, borders, box shadows, and text. The data.js file contains a list of all the songs in the music player, with the song name, file path, artist name, and cover image. The app.js file contains all the variables and methods for the music player to work. It controls the play/pause, forward, backward, and shuffle buttons and matches the slider with the current time and the song duration.

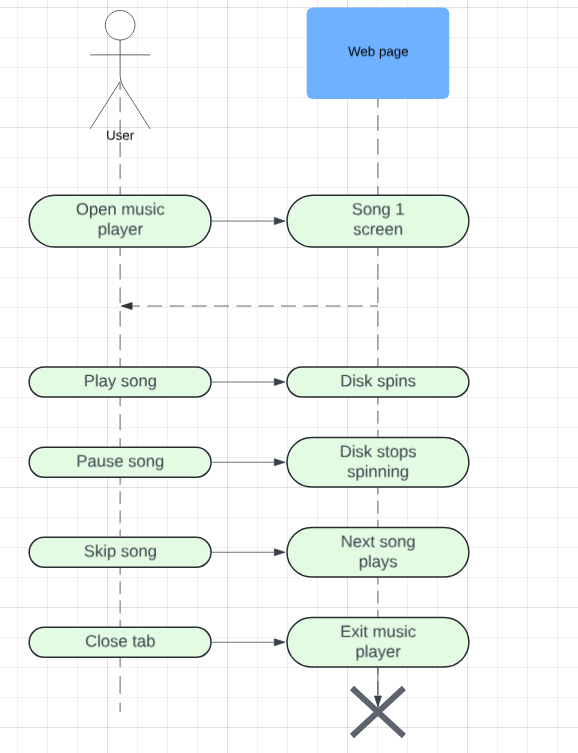
Repository: <https://github.com/ekkanikanti/music-player.git>

**Static View**



The **index.html** contains the physical elements of the music player and controls how those elements appear on the screen. The layout and appearance is controlled by **styles.css** and the actions and events are controlled by the **app.js** file. The JavaScript files contain the functionalities that allow the disk to rotate and the user to play and pause songs. The **data.js** file contains a list of songs with the song name, file path, artist name, and cover image path. The **app.js** file takes in the list and allows the user to switch between songs using the forward and backward buttons or shuffle the list of songs with the shuffle button. It uses a for-loop and math.random to randomize the list of songs in **data.js**. Clicking the next or previous button after the list has been shuffled will play the next or previous song in the randomized list. The **index.html** allows the cover image, text elements, and buttons to appear on the screen so the user can listen to music and see what song is playing.

**Dynamic View**



When the user opens the **index.html** file, their default web browser will open and they will see a music player on the screen. The first song in the list will be shown on screen, with the name, artist, album cover, song duration, current song time, slider, play button, shuffle button, and the forward and backward buttons. If the user clicks the play button, the song audio will play and the disk will spin, and the button becomes a pause button. When the pause button is clicked, the song audio will stop and the disk will stop spinning. If the user clicks the next or previous buttons, the next song or the last song will be played and the name, artist, album cover, and song duration will be updated accordingly. If the user clicks the shuffle button, the list of songs in the data.js file will be shuffled and a random song will play on the screen. If the user closes the browser tab, the music player will also close.

**Design Rationale**

The decision to build the music player using only frontend technologies (HTML, CSS, JavaScript) is driven by the following factors:

* **Simplicity**: The absence of a backend reduces complexity, making the music player easy to deploy and maintain
* **Offline Access**: By avoiding server-side dependencies, users can listen to music offline without needing internet access
* **Cost-Efficiency**: Since the solution does not require any server infrastructure, it is entirely free to deploy and run

However, the design does have its limitations:

* **Scalability**: While this device works for single-device offline use, syncing songs across devices or more advanced features like multi-user support would require a backend
* **User Preferences**: There is no user data storage due to only local storage options, which is not shareable across devices. This is a trade-off for the simplicity of the frontend only approach