- **How will your server use threads/goroutines? In other words, when are they created, and for what purpose?**

Whenever a new client connects to the server, the server creates a new thread for this client. This is to keep each client's socket data (control changes, streamed music data) independent of one another. The server should also create a thread for each station it has. Each station can thus independently broadcast different data concurrently.

- **What data does your server need to store for each client?**

The server needs to keep track of each client's thread ID (to gracefully handle unexpected client shutdowns); socket number, address, and UDP listening port (to be able to stream data to the correct location); and the station the client is connected to (could possibly just store this information in a client list for each station though).

- **What data does your server need to store for each station?**

Each station must stream a particular song indefinitely. So, each station must store/have access to its corresponding song file. Each station also needs to store a list of clients that are connected to it per the specs.

- **What synchronization primitives (channels, mutexes, condition variables, etc.) will you need to have threads/goroutines communicate with each other and protect shared data?**

The client lists for each station must be protected by mutexes. Without mutexes, multiple clients concurrently requesting to switch to/leave a particular station will lead to race conditions.

- **What happens on the server (i.e., how does the stored server state change) when a client changes stations? (Thinking about this may help you answer the previous question.)**

Since the server needs to keep track of which clients are listening to which stations, the server (possibly the individual station thread) must remove the client from its list of connected clients and add that client to the new station's client list.