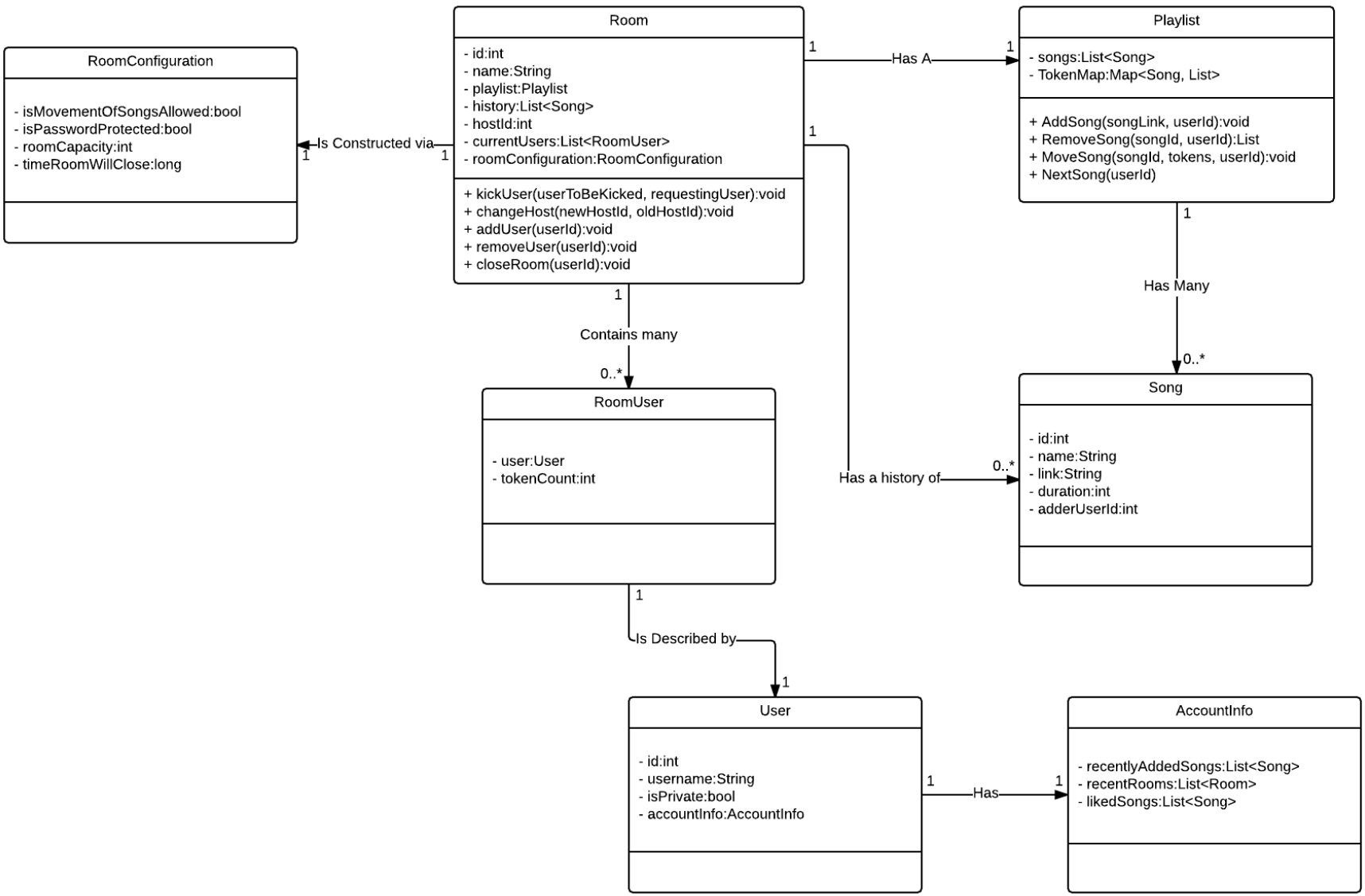
RoomConfiguration
- isMovementOfSongsAllowed:bool
- isPasswordProtected:bool
- roomCapacity:int
- timeRoomWillClose:long
Is Constructed via
1
1
Room
- id:int
- name:String
- playlist:Playlist
- history:List<Song>
- hostId:int
- currentUsers:List<RoomUser>
- roomConfiguration:RoomConfiguration
+ kickUser(userToBeKicked, requestingUser):void
+ changeHost(newHostId, oldHostId):void
+ addUser(userId):void
+ removeUser(userId):void
+ closeRoom(userId):void
1
Has A
1
Playlist
- songs:List<Song>
- TokenMap:Map<Song, List>
+ AddSong(songLink, userId):void
+ RemoveSong(songId, userId):List
+ MoveSong(songId, tokens, userId):void
+ NextSong(userId)
1
Has Many
0..*
1
Contains many
0..*
RoomUser
- user:User
- tokenCount:int
1
Has a history of
0..*
Song
- id:int
- name:String
- link:String
- duration:int
- adderUserId:int
1
Is Described by
1
User
- id:int
- username:String
- isPrivate:bool
- accountInfo:AccountInfo
1
Has
1
AccountInfo
- recentlyAddedSongs:List<Song>
- recentRooms:List<Room>
- likedSongs:List<Song>

The classes in the diagram represent the model for our application. They will be constructed from database tables, and then passed to the front-end for viewing. All private attributes of the objects are assumed to have getters and setters. They were left out to save space.

Most of the objects do not have methods, as they are data containers. After construction, they are simply there to be viewed by a client.

The *Room* class represents an active instance of a room. Users can connect to it, view its playlist and history, and look at the other users currently in the room. *Rooms* will be constructed with the *RoomConfiguration* object, which will be populated on the web-page by a user.

The *User* object contains all the information that we store about a user. The *AccountInfo* object is kept separate to allow for lazy-loading from the database, as well as denying access if the user account is marked private. *RoomUser* is a runtime container for user and tokens. Tokens can be used in a room to let users move songs up in a playlist.

The *Playlist* class stores a list of *Songs* as well as a *Map* object. The *Map* object keeps track of what songs currently have tokens applied to them, so that they can be refunded to the correct user once the song is played.