Team 6: Jinghu Lei, Lainey Hall, Jason Mayfield

**Cache**

For user authentication, we can use OAuth and browser cookies to let a user “stay” logged in without user interaction after the first time. For our database, we should also cache the user’s custom preferences that are generated with our application on each instance as well as their own Spotify affinity that we can cache.

* For user preferences:

We would cache the different genres/categories that a user likes and save them after each song to the database. When the application is exited and starts up again, we can just load the users past preferences using their profile id. This would make it a seamless experience across each time they open our application and across different browsers; their preferences would be accessible anywhere. Correspondingly, we will use the same preferences to get recommendations from Spotify and give those to the user.

* For user affinity:

Initially, we will use Spotify’s recommendation system with random genres to fetch some starting songs to test on the user. This allows us generate some preferences with this information to cache song and artist recommendations for the user. Then, we can start caching those songIDs and artistIDs before they even start playing a song. This would make their loading experience much faster between songs. This is the make up of the *queue* property for the user’s playlist. It stores the songs they like as well as a queue of upcoming songs.

**Data Models**

const PreferenceSchema = new Schema({

name : {type : String, required : true}, *// name of the category*

id : {type : String, required : true}, *// categoryID, required – maps to the genre cache*

type : {type : String, required : true} *// the type of category e.g. artist, genre, track*

});

const UserSchema = new Schema({

name: String, *// user name, required*

id : {type : String, required : true}, *// user id from Spotify, required*

playlist\_uid : Schema.Types.ObjectId, *// custom playlists for our app*

preferences : [PreferenceSchema] *// preference array based on category*

});

const SongSchema = new Schema({

name : {type : String, required : true}, *// name of the song*

id : {type : String, required : true}, *// songID from Spotify, required*

uri : {type : String, required : true}, *// song uri for playback*

});

const PlaylistSchema = new Schema({

id : {type : String, required : true}, *// id of the user playlist*

songs : [SongSchema], *// songids for spotify*

queue : [SongSchema] *// queue of songs to add*

});

Correspondingly to the Mongo object schema, we would have two JSON documents. The first being a User JSON with embedded preferences. The second being our category cache, with songs embedded in those.

**User JSON**:

{

“name” : “John Doe”,

“userid” : “1234567”,

“playlist\_uid” : “3ba89acs98”,

“preferences” :

[

{

“name” : “The Beetles”,

“id” : “ab1823eca”,

“type” : “artist”

},

{

“name” : “Country”,

“id” : “country”,

“type” : “genre”

},

{

“name” : “Hey Jude”

“id” : “1234kals”,

“type” : “track”

}

]

}

**Playlist JSON**:

{

“categoryid” : “toplists”,

“songs” :

[

{

“name” : “Happy Now”,

“songid” : “4keoy2fqgwGnbWlm3ZVZFa”,

“artistid” : “2qxJFvFYMEDqd7ui6kSAcq” // zedd

},

{

“name” : “I Love It (& Lil Pump)”,

“songid” : “4S8d14HvHb70ImctNgVzQQ”,

“artistid” : “5K4W6rqBFWDnAN6FQUkS6x” // kanye

}

]

}

**Sequence Diagram – Also Attached**

