Project Database

The project’s backend is powered by Google Firebase. For database, Cloud Firestore is preferred over Realtime Database due to its structure, functionality and regular development by the Google team. This database as any Firestore database uses a Collection-Document model. The database is designed as such to include fields in the documents to reduce read-time as much as possible and improve searchability. This is done with an emphasis to improve user experience so that all heavy work is done at the time of database design and writing as document read instances far outnumber the document write instances.

The Collections in the Database are outlined below:

* Users: Collection to store the information of the signed-in users.

Document ID: Random

Document fields:

{

‘email’: Email used by the user to register

‘username’: Unique username chosen by the user at signup time

‘fullname’: Full name of the user

‘searchArray’: Array containing substrings of the ‘username’ and ‘fullname’ to be utilized in the search functionality

}

* FriendPairs: Collection with each document containing information on a friendship pair.

Document ID: Random

Document fields:

{

“friend1”: username of User sending the request

“friend1name”: fullname of user sending the request

“friend2": username of user receiving the request

“friend2name”: fullname of user receiving the request

“status": flag indicating state of friendship. ‘pending’ indicates request sent but not accepted. ‘active’ indicates accepted request.

}

* Parties: Collection with each document with information of the party group.

Document ID: username of the creator

Document fields:

{

'creator': username of user creating the party

'creatorName': fullname of user creating the party

'memberCount': current number of users who joined the party

'member': array with the username of all the user in party

'memberName': array with the fullname of all the user in party

'searchArray': array with substrings of the creator’s username and fullname to implement search functionality

'partyStarted': status indicating whether party has started(‘yes’) or is accepting new joiners(‘no’)/ is searchable

‘docRef’: document reference to the movie reference list to be used to feed the random movies to users

'collectionRef’: name of the collection from which movie data is to be downloaded.

‘member[i]’ : array containing likes or dislikes in the party game of member[i]

}

* {MovieCollection} (Actual collection name depends on the grouping): Collection with each document with the movie data.

Document ID: Random

Document fields:

{

‘id’: unique ID of the movie (created before uploading the data)

‘title’: title of the movie

‘duration’: duration of the movie

‘year’: year of release

‘synopsis’: brief synopsis of the movie

‘image’: URL of the image of the movie cover

‘genre’: genre of the movie

‘platform’: platform in which movie is available

}

* MovieRef: Collection with each document having the list of all document references of a {MovieCollection}.

Document ID: Collection name of the MovieCollection for which the doc has references

Document fields:

{

‘docRefs’: an array with all the document references to all the documents in a MovieCollection.

}

* ActiveParties: Collection which is contains the match status of the card game. Document in this function are created and deleted by Cloud functions triggered by creation or deletion of Parties doc.

Document ID: username of the creator

Document fields:

{

'docRefs': Array containing the doc references of the randomized movies

'creator': username of the creator of the party

'match': status of the game. ‘yes’ for a match & ‘no’ for not matched yet

'matchRef': the array position of the matched movie

‘collectionRef’ : Name of the collection of the movie from which random collection is made.

}