DAS CORE PRINZIP

INFORMATION SYSTEM

M.Sc. Big Data & Business Analytics

Project Report on VIRTUAL TOURIST

Narendra Medisetti

HOCHSCHULE SRH HEIDELBERG
Intelligence in Learning

STAATLICH ANERKANNTE HOCHSCHULE

1 Introduction

In today’s busy world, everybody adores the idea of visiting on an exotic travel destination, however not every person has the time and willingness to go through the discomforts that comes during the tour. From the difficulties and worries of the actual travel experience to the time required, sometimes the actual goal of traveling isn't exactly conceivable. This is how the concept of virtual tourist came into existence. As the technology progresses, more travel destinations are finding their ways with the help of web and the best virtual reality 3D technology to provide maximum number of people, have a close to real life experience from any smartphone, tablet or PC connected to the internet without leaving their ground. Virtual tourist has its applications in various domains such as real estate properties, cottages, hotels, tourist sites, vehicles, businesses, industries and showrooms. However, we have limited the scope of our project to virtually
experiencing a live music concert.

Objective of this project is to revolutionize the idea of visiting the music concerts or music festivals. It is a new way of experiencing a live concert. Since cameras are positioned onstage as well as in the audience, artists who is performing and attendees who are attending the concert can have their respective point of view in the concert. If developed further and implemented VR technology as well, viewers will be able to feel what exactly it feels like to be on the stage and in the audience. This all can be experienced seated in your own confirmable chair at home. This also allows the user to be able to be transported to the old concerts, which the person has actually visited in the past or missed out on the opportunity. Using the 360-degree-view, one can look around and see other people and interact with them.

The scope of our project is limited to creating the database required for storing data related to all the concerts, artists and attendees and fetching the required data based on defined user stories. Our database is implemented in a new NOSQL database i.e. Arango DB, it offers a built-in support for graphs along with a document store. It allows to structure the data as a graph with vertices and edges with providing the flexibility of number of attributes. On the stored data, queries can be fired with a similar language to SQL called AQL. To ease the uploading of binary data files such as audio, video and image files, an external application called “Foxxy” makes use of Foxx services in ArangoDB and provides an interface for the same.

2 Organization

2.1 Responsibility in project and report writing

Jasmine	**Ali**	**Anish**	**Komal**	**Narendra**
User Story 1	User Story 2	User Story 3	User Story 4	User Story 5
Quality Control	Team Leadership	Strategic Influencing	Team Leadership	Data Extraction
Data Modeling	Team Organization	Foxxy Implementation	Foxxy Implementation	Data Preprocessing
Evaluation Check	Keeping track of process	Design Front End and connect with ArangoDB	Data Preparation	Quality Control
Documenting and Report Editing	Quality Control	Creating APIs	Creating graphical representation for the report	Query Design
Chapter 3. Use Case and Chapter 4. Database	Chapter 2. Organisation and 7. Queries	Chapter 6. Implementation and Chapter 8. evaluation	Chapter 1. Introduction and Chapter 5. Data Model	Chapter 8. evaluation and Chapter 9. Bibligrapy

To organize the teamwork effectively is very important to make best use of every individuals knowledge and potential. In the beginning of this project, for finalizing the user stories, we encouraged all the team members to be physically present and actively participate in the respective discussions. Everyone brainstormed different possible ideas for user stories to discuss and estimate the feasibility. After thorough discussion and considering everyone's opinions, we finalized the User stories and organized the team meetings thrice a week ranging over four months. Once everyone got the idea of how to proceed with the implementation, gauging individuals' capability we assigned the work. For the updating and tracking everyone's work progress, we used a tool called Trello. This tool turned out to be very helpful for us to plan our project tasks on a single platform. It helped us to update and be in the sync when somebody updated their given task. This tool has a color labeling feature to assign priority to certain tasks, to draw attention of everyone and to follow the decided timeline to avoid delays in any tasks. For communication and sharing documents we used made use of a team collaboration tool called Slack.

3 Use case in detail

3.1 User story 1: “John is an artist who plays EDM Music, he wants to look for concerts where he can play his music along with the date and time of the concert virtually.”

Concerts are the place where artists are recognized through their music and every artist needs a loving crowd who can enjoy their music. So, this feature provides the virtual tour of different concerts and artists along with the date and time and stage of the performance.

Requirements:

- Concerts Data
- Artists Data

3.2 User Story 2: “David is a music lover who wants to virtually attend the concert Ultra Music Festival and enjoy the drinks along with interacting with other attendees.”

This feature would help the spectator or attendee to have the real experience without leaving his couch. With this he will be getting a virtual tour of the entire venue that includes all the stages, food & beverage counters and a glimpse of the festive atmosphere through videos and images and also the kind of music played in the concert. He could have the option of interacting with another attendee like him.

Requirements:

- Concerts Data
- Attendees data
- Food & Beverage

3.3 User Story 3: “Kevin is a pop music fan who would like to virtually go back in time to the Shawn Mendes Concert in 2018 and connect it through videos and lyrics.”

There are some concerts which leaves a mark or a memory in a person’s heart. Which they want to visit again and again. So, this feature lets the attendee to visit the concert happened in past and experience the same virtually over this application.

Requirements:

- Artists
- Past Concerts
- Media link

3.4 User Story 4: “Thomas is virtually watching a Martin Garrix concert show in an AUDI2, wants to have the option to look for another show in different auditoriums in a concert of all the possible artists/bands which has higher number of viewers on that particular day with time.”

Every person has its own liking for music and concerts, but there are times when a person wants to go for a concert where there are maximum numbers of attendee's. So, this feature allows an attendee to look for concerts happening in different auditoriums where there are maximum crowd.

Requirements:

- Number of Attendee's in a concert auditorium
- Artist Data
- Concerts Data

3.5 User Story 5: "Michael is an advertiser who would like to know advertising possibilities in Ultra Music festival based on availability of slots and quotation for the same along with the slot location in concert."

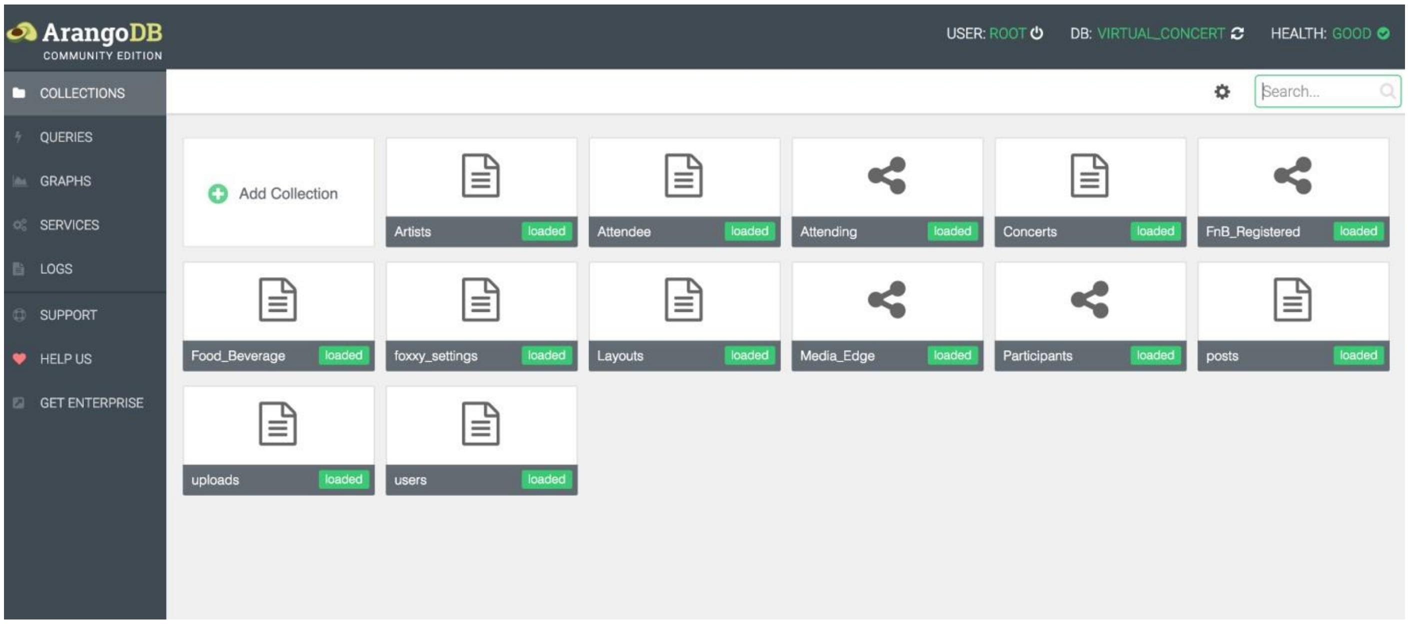
In any concerts there are several things to be covered, one of it is the advertisements which is nothing but the sponsors way to attract public towards their Advertisements. So, this feature allows the advertisers to look for available slots in the concert layout to put up their advertisements boards which may catch more of public eyes.

Requirements:

- Layout of Concert
- Name of Concert
- Number of slots available to book

4 Databases

ArangoDB is a NoSQL Database which is multi- model memory database. In this database, the data can be store in more flexible way such as documents, key-value pair or graphs; it also has single query language which is ArangoDB query language (AQL) through which different models can be combined in single query.

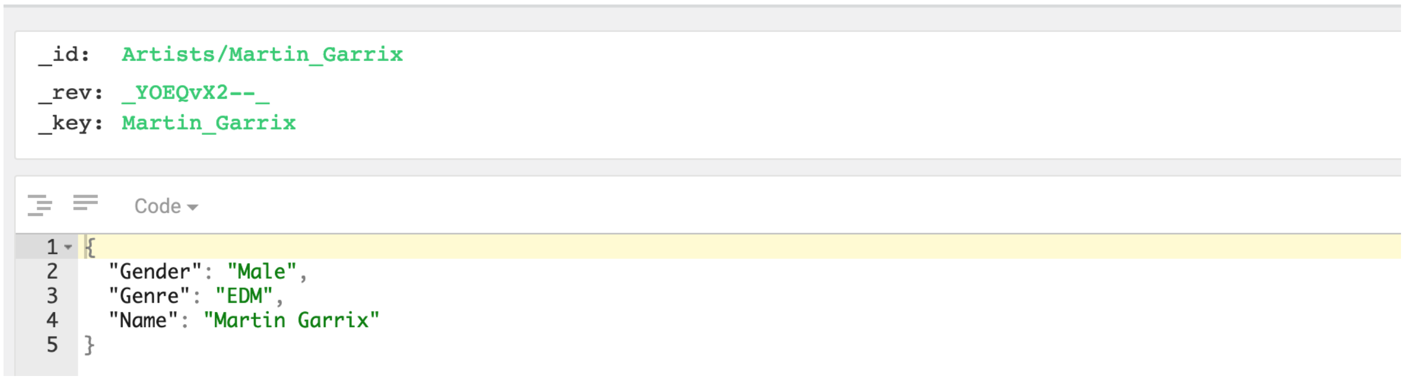


Various Data models are supported by Arango DB:

4.1 Document Model

A document-oriented database is used for storing, retrieving and managing document-oriented information, such as semi-structured data. We can also import/export the data as JSON files.

Examples: Attendee document, Concerts and Artists Documents has been created in our project.



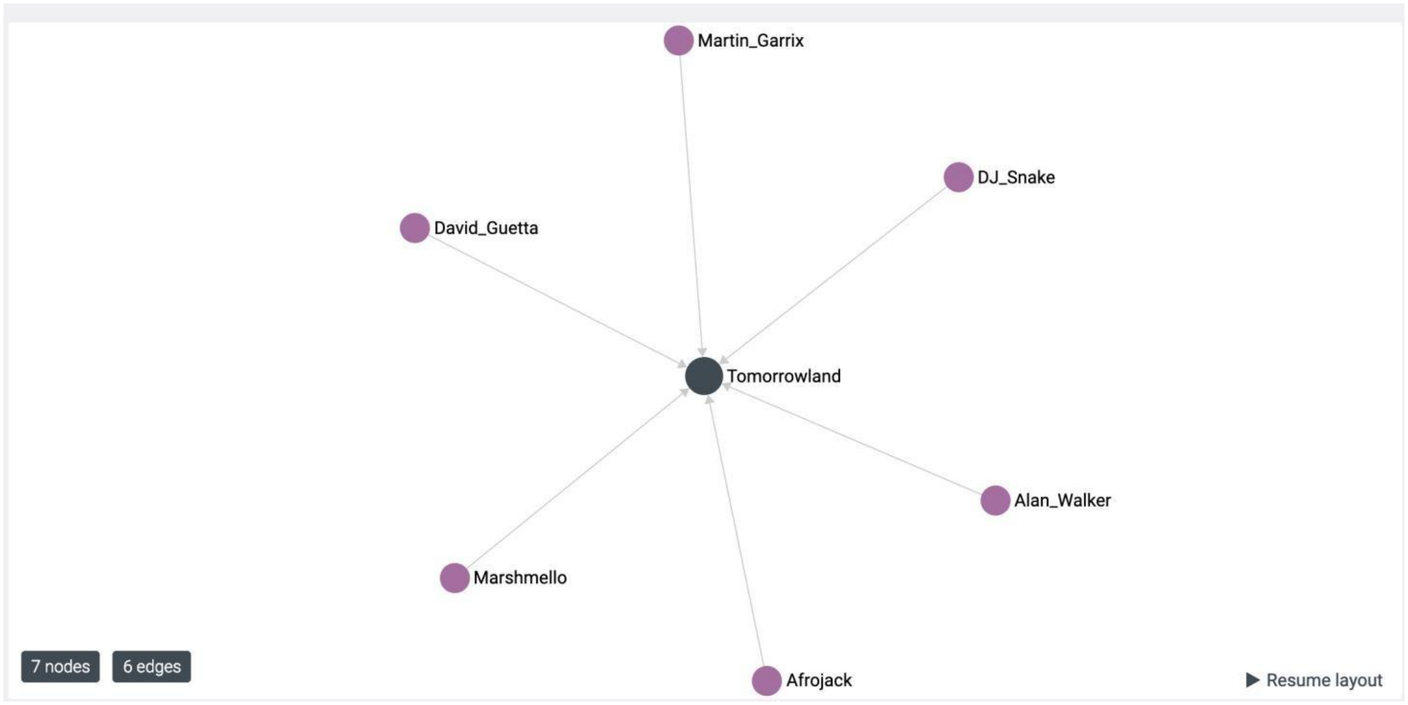
```
_id:   Artists/Martin_Garrix
_rev:  _YOEQvX2--_
_key:  Martin_Garrix
```

```
{
  "Gender": "Male",
  "Genre": "EDM",
  "Name": "Martin Garrix"
}
```

4.2 Graph Data model

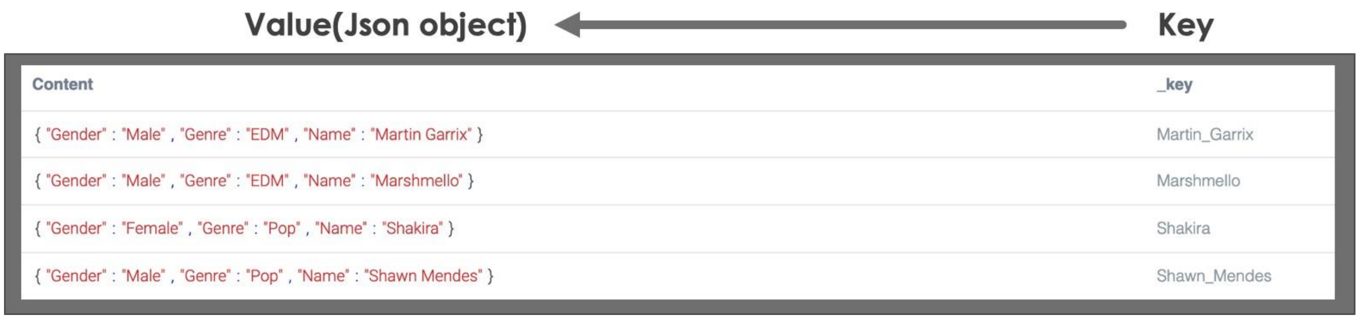
A graph database is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data. It directly relates data items in a tore a collection of nodes of data and edges representing the relationship between the nodes.

Examples: Artist to concerts, Attendees to concert



4.3 Key/Value Store Model:

Map value data to unique string keys that are used as identifiers. This helps in implementing scaling and partitioning easily due to simplistic data model. Key-value can be seen as a special case of documents. For many applications this is sufficient, but not for all cases.

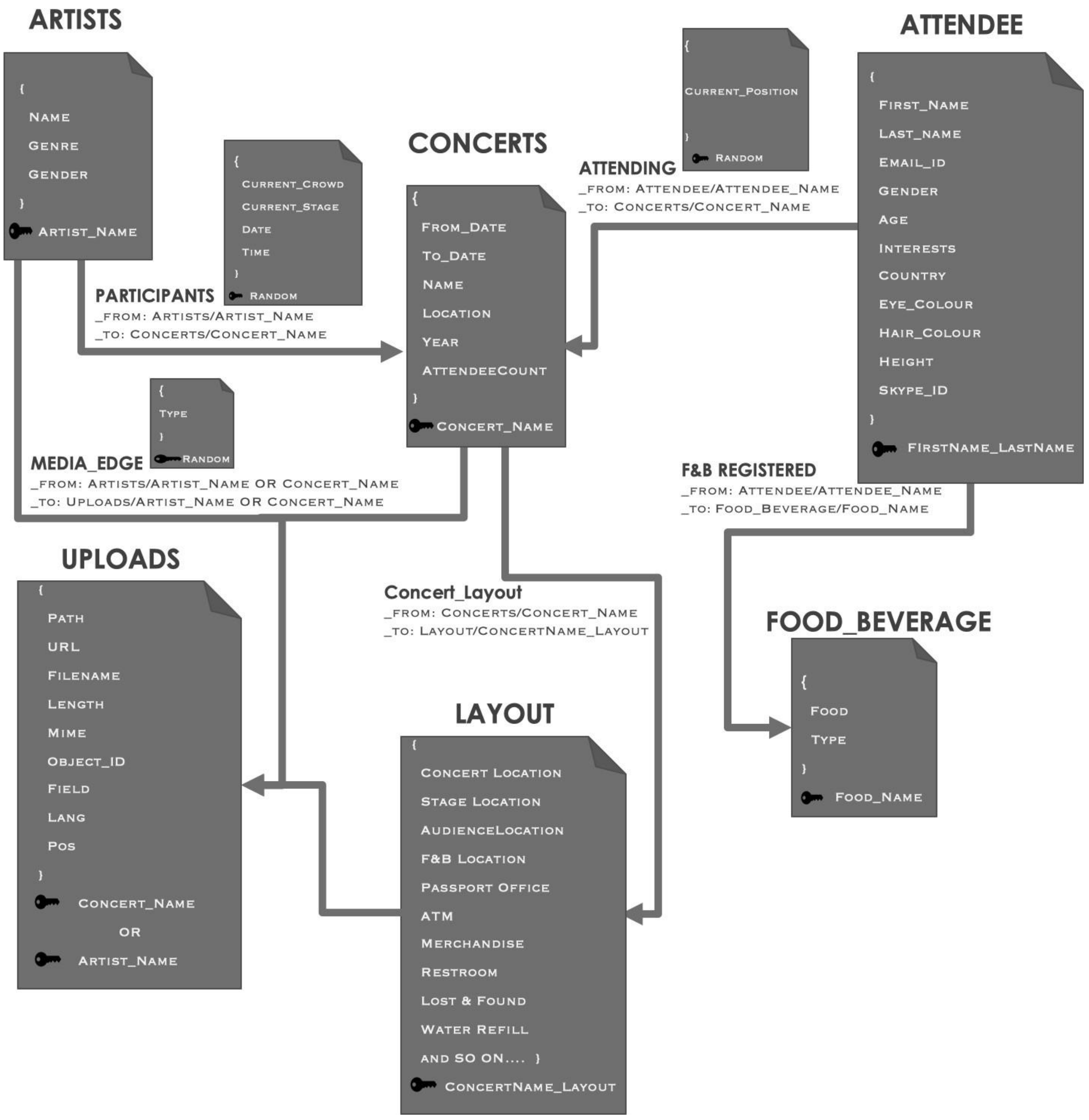

Value(Json object)
Key

Content	_key
{ "Gender" : "Male" , "Genre" : "EDM" , "Name" : "Martin Garrix" }	Martin_Garrix
{ "Gender" : "Male" , "Genre" : "EDM" , "Name" : "Marshmello" }	Marshmello
{ "Gender" : "Female" , "Genre" : "Pop" , "Name" : "Shakira" }	Shakira
{ "Gender" : "Male" , "Genre" : "Pop" , "Name" : "Shawn Mendes" }	Shawn_Mendes

5 Data Models

5.1 Mapping of requirements to data models

To suffice the challenges of effectively storing and querying the data, in general a document store is being used for storing the unstructured data, a graph database is used to leverage the highly linked referential data and a key-value store is used for storing the data as a hash table which has a unique key pointing to a specific object of the data. In order to learn the working of a multi-model database and to have the flexibility of expanding the data models to their fullest extent, Arango DB is used for this project. As per the requirements of our User stories, we require to store all the data related to different Artists, Concerts and Attendees. Also, we need to define a specific relationship between these entities to retrieve the respective attributes to meet our user stories requirements.

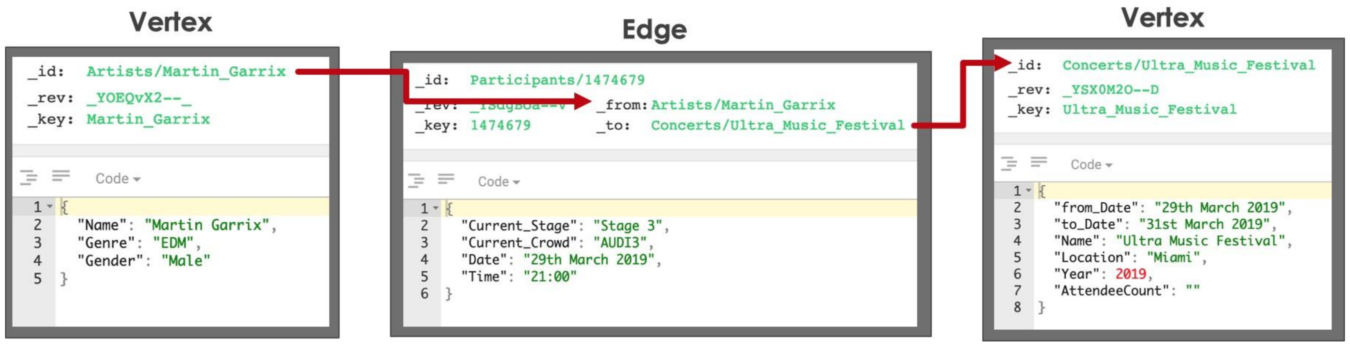


ARTISTS
{ Name Genre Gender } Artist_Name
{ Current_Crowd Current_Stage Date Time } Random
PARTICIPANTS
_from: Artists/Artist_Name
_to: Concerts/Concert_Name
CONCERTS
{ From_Date To_Date Name Location Year AttendeeCount } Concert_Name
{ Current_Position } Random
ATTENDING
_from: Attendee/Attendee_Name
_to: Concerts/Concert_Name
ATTENDEE
{ First_Name Last_Name Email_ID Gender Age Interests Country Eye_Colour Hair_Colour Height Skype_ID } FirstName_LastName
{ Type } Random
MEDIA_EDGE
_from: Artists/Artist_Name OR Concert_Name
_to: Uploads/Artist_Name OR Concert_Name
F&B REGISTERED
_from: Attendee/Attendee_Name
_to: Food_Beverage/Food_Name
UPLOADS
{ Path URL Filename Length Mime Object_ID Field Lang Pos } Concert_Name OR Artist_Name
Concert_Layout
_from: Concerts/Concert_Name
_to: Layout/ConcertName_Layout
LAYOUT
{ Concert Location Stage Location AudienceLocation F&B Location Passport Office ATM Merchandise Restroom Lost & Found Water Refill and so on.... } ConcertName_Layout
FOOD_BEVERAGE
{ Food Type } Food_Name

As shown in the above figure we have stored the data of Artist, Concerts, Attendees, Food & beverage, Uploads and Layouts in the document collections. In Arango dB every document has a unique primary key i.e. "_key attribute" which identifies the different documents in that specific Document collection. This can be specified as key-value store. Here keys are stored in the string format and the values are JSON documents. These JSON documents allows the flexibility of the documents in the database. The data in Uploads document collection comprises of meta data of images, audio and video files which is populated with the Foxxy framework from where we have uploaded the media files.

To define the relationships between these document collections, edge collections such as participants, Media_Edge, Attending, and F&B Registered are defined. Here, the edge collections are also combination of JSON documents but with two extra mandatory attributes, "_from attribute" and "_to attribute", which refers to the start and end of the vertices of the edge. This is the implementation of graph model where Artists, Artist, Concerts, Attendees, Food & beverage, Uploads and Layouts are the vertices and participants, Media_Edge, Attending, and F&B registered are the edges. These relationships keep the data to be linked directly with each other and helps to access desired data with one single query.

In Arango DB, every document is assigned with a unique document handle i.e. "_id attribute". It is a string and is combination of collection's name and the document key i.e. "_key attribute" which separated by "/". Having unified the query language AQL for this multi-model database, it allows the users to traverse through one document to another as shown in the below diagram.



```
_id:   Artists/Martin_Garrix
_rev:  _YOEQvX2--_
_key:  Martin_Garrix

{
  "Name": "Martin Garrix",
  "Genre": "EDM",
  "Gender": "Male"
}
```

```
_id:   Participants/1474679
_rev:  _ISugBOa--V          _from: Artists/Martin_Garrix
_key:  1474679              _to:   Concerts/Ultra_Music_Festival

{
  "Current_Stage": "Stage 3",
  "Current_Crowd": "AUDI3",
  "Date": "29th March 2019",
  "Time": "21:00"
}
```

```
_id:   Concerts/Ultra_Music_Festival
_rev:  _YSX0M2O--D
_key:  Ultra_Music_Festival

{
  "from_Date": "29th March 2019",
  "to_Date": "31st March 2019",
  "Name": "Ultra Music Festival",
  "Location": "Miami",
  "Year": 2019,
  "AttendeeCount": ""
}
```

6 Implementation

This section describes the implementation of each of the above-mentioned data models. It also describes the dummy data used for your use case.

ArangoDB Installation:

To install ArangoDB Community Edition on the system visit this link: https://medium.com/@pranshumalviya/graphical-interpretation-of-data-using-arangodb-8d8e5ef7c622

ArangoDB is available for various environments that includes both MAC and Windows operating system as well.

Installation in MAC

It can be installed using homebrew by using the following command.

```
brew install arangodb
```

This command installs the most recent stable version available for ArangoDB and its dependencies.

Server is installed in

```
/usr/local/sbin/arangod
```

To start the ArangoDB server, use below command

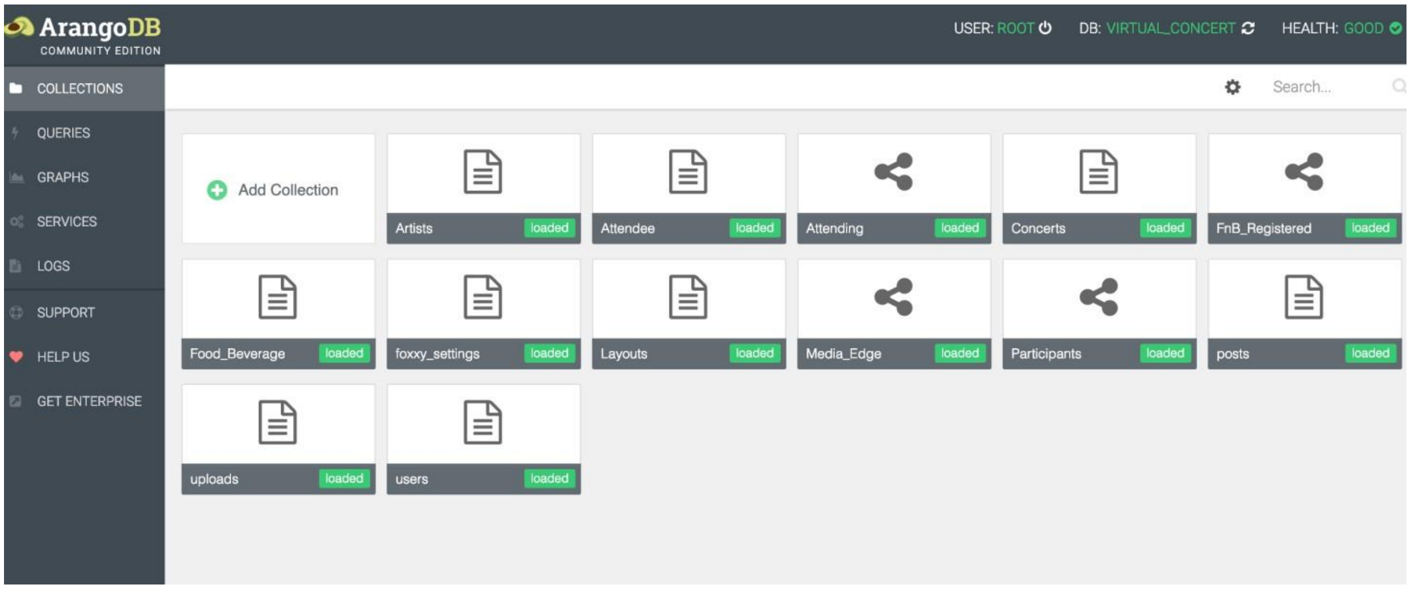
```
/usr/local/sbin/arangod &
```

The ArangoDB server viz “Arangod” accompanies an implicit web interface for organization. It allows the user to handle databases, collections, services, graphs and also provides the overall dashboard of the current database in use. This can be used to run the required queries in a very simple manner. Insights and server status are also given on dashboard.

The Web Interface which is also known as Web UI, frontend or Aardvark, can be reached by accessing the URL **http://localhost:8529** with default server settings. Since the Web Interface is easier to use than the Arango Shell (arangosh), we have therefore made use of the Web UI for creating the database, collections and also for inserting documents.

Arango database is group of collections. Collections are accumulations of records which are known as documents. By default, the Arango Web UI makes use of the _system database for the creation of collections and documents.

We have created a database named Virtual_Concert to serve our purpose.



Type of Collections

1. Document Collection:

This is the default type of collection which stores the all the documents with _key as the unique key.

-key attribute can be defined by users when they save their documents this key will be saved as a unique attribute along with the respective document. Users need to define the key by following certain naming conventions.

1. key must be a string value starting with an alphabet or a special character but not a digit.

2. It cannot be empty or containing white spaces

3. It is case sensitive

2. Edge Collection:

Edge collections are specific collections that store edge documents which are associated documents that refer to different documents. This sort of collection must be determined when the collection is created and can't be changed a while later.

To change edge endpoints, you would need to delete the previous document/edge and insert again. Other fields can be changed as per the requirement like a normal collection.

Edges are the documents which always comprise of “_from” and “_to” attribute other than “_key”.

_**from**: This is the document handle of the linked vertex which can be also referred as “incoming relation”

_**to**: This is the document handle of the linked vertex which can be also referred as “outgoing relation”

JSON objects are used to store documents in arangoDB. These objects can also be nested to any depth. Each object is uniquely assigned to a document handle which is a unique key as mentioned above.

In Arango dB Web UI we can create documents in following two ways

1. Using ArangoDB Web UI editor

once you create a document it prompts you to enter a unique key for the document. If the unique is not entered, Arango dB autogenerate the unique key. After which, the use can store the document in by manually typing a JSON object.

2. Using Upload option

This feature allows the use to upload an entire JSON object in the form of .json file. Once the file is uploaded all the documents appear in the collection.

3. Using Queries

By using the INSERT command through AQL queries, we can insert new documents into the specified collection.

As per the project specifications it is required to store binary data in the form of images and videos. The support binary data storage and big data blobs are being added on the next release of Arango DB. Hence, we have to resort to the usage of Foxx services. We can create a Foxx service to create a file in our system and reference that file name inside our ArangoDB database.

Foxxy is a tool which generates the code that is useful for creating an application that could be used to upload audio and video files by using ArangoDB, Foxx, UIKIT3, RIOTJS and brunch. The primary vision of foxxy is to aid you in creating an application quickly and expose restful API through Arango DB’s Foxx tool. This method simplifies the way of implementing a service to upload binary files from the Arango web UI.

Foxxy Installation

Requirements:

- OSX Or Linux device
- NodeJS & NPM + yarn
- ArangoDB

Steps to perform for creating a Foxxy App

1. To install Node, NPM and yarn

```
brew install node
npm install -g yarn
```

2. To install foxxy

```
npm install -g foxxy
```

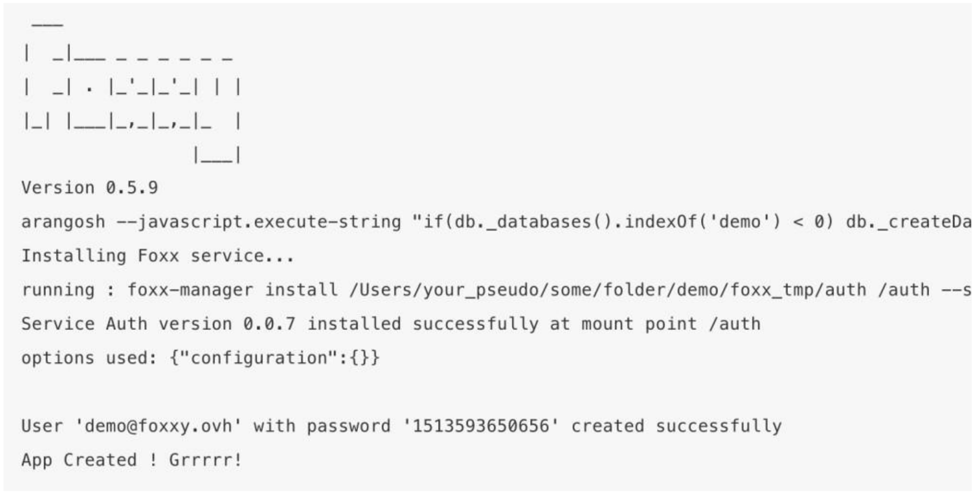
3. To create a new application

```
foxxy new demo --database demo
```

In our case we used the command

foxxy new Gallery -–database Virtual_Concert

after running above command, we will get below output.



```
 ___
|  _|___ _ _ _ _ _ _
|  _| . |_'_|_'_| | |
|_| |___|_,_|_,_|_  |
                |___|
Version 0.5.9
arangosh --javascript.execute-string "if(db._databases().indexOf('demo') < 0) db._createDa
Installing Foxx service...
running : foxx-manager install /Users/your_pseudo/some/folder/demo/foxx_tmp/auth /auth --s
Service Auth version 0.0.7 installed successfully at mount point /auth
options used: {"configuration":{}}

User 'demo@foxxy.ovh' with password '1513593650656' created successfully
App Created ! Grrrrr!
```

4. Yarn is used to install js dependencies which can be installed with below command

```
cd demo; yarn install
```

In our case

Cd Gallery; yarn install

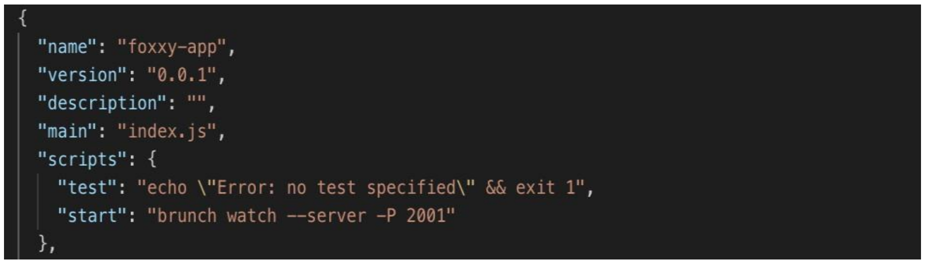
5. To add a new CRUD to foxxy application

```
foxxy g crud post
```

The same command also creates the **Uploads** widget in the application

6. Modify default application port

   - Open the package .json file in gallery folder and modify the port number according to your choice

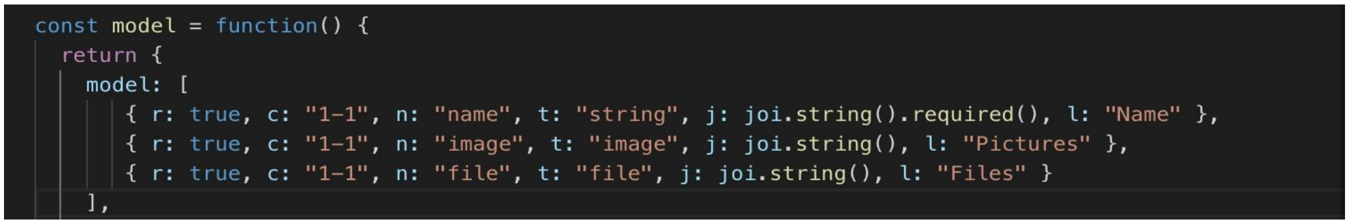


```
{
  "name": "foxxy-app",
  "version": "0.0.1",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "brunch watch --server -P 2001"
  },
```

7. To define the post model

After the creation of CRUD, you must modify your foxx/cruds/APP/models/post.js file.

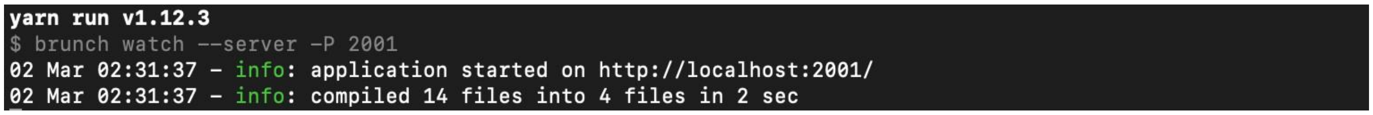
Fill the return array with the fields definition which you would like to incorporate in your upload widget application



```
const model = function() {
  return {
    model: [
      { r: true, c: "1-1", n: "name", t: "string", j: joi.string().required(), l: "Name" },
      { r: true, c: "1-1", n: "image", t: "image", j: joi.string(), l: "Pictures" },
      { r: true, c: "1-1", n: "file", t: "file", j: joi.string(), l: "Files" }
    ],
```

8. To start the application

```
yarn start
```

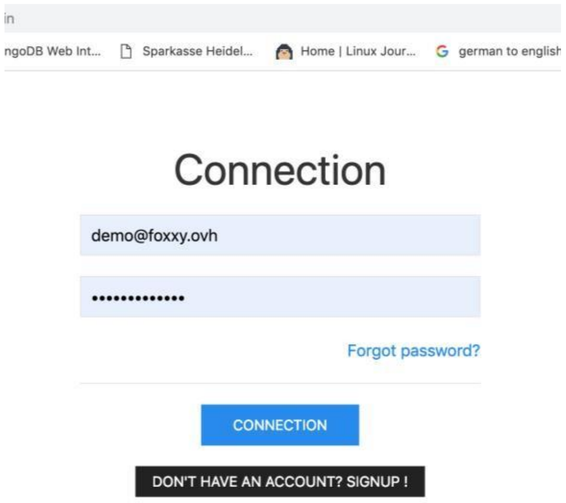


```
yarn run v1.12.3
$ brunch watch --server -P 2001
02 Mar 02:31:37 - info: application started on http://localhost:2001/
02 Mar 02:31:37 - info: compiled 14 files into 4 files in 2 sec
```

The application is running on the localhost with port 2001.

9. Launch the Foxxy application and login with the generated credentials during creation.

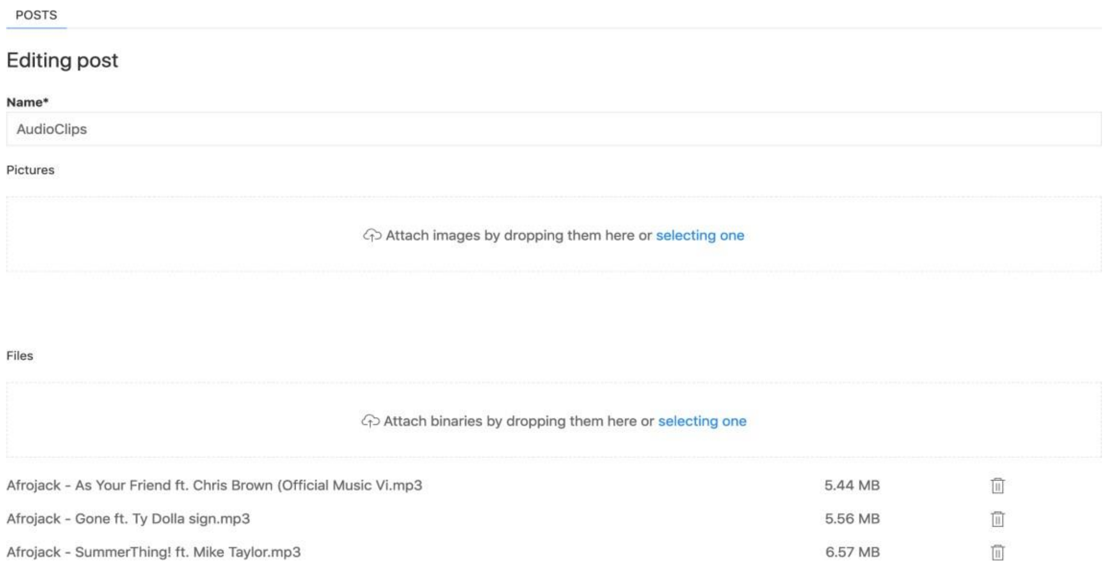
10. Create a new post and edit it to the files or images.



Connection

demo@foxxy.ovh

•••••••••••••



POSTS
Attach images by dropping them here or selecting one
Attach binaries by dropping them here or selecting one

Editing post

Name*

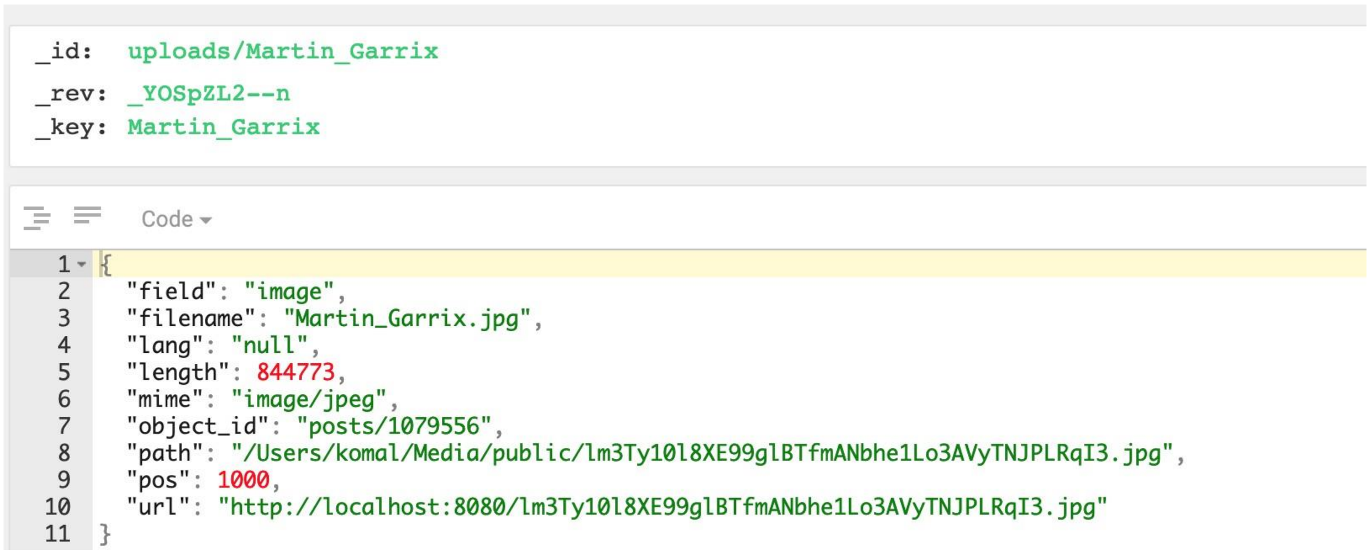
AudioClips

Pictures

Files

Afrojack - As Your Friend ft. Chris Brown (Official Music Vi.mp3	5.44 MB
Afrojack - Gone ft. Ty Dolla sign.mp3	5.56 MB
Afrojack - SummerThing! ft. Mike Taylor.mp3	6.57 MB

Once the files are successfully uploaded in the local, a new collection called uploads is created in the database which contains the metadata of the uploaded files.


```
_id:  uploads/Martin_Garrix
_rev: _YOSpZL2--n
_key: Martin_Garrix
```

```
{
  "field": "image",
  "filename": "Martin_Garrix.jpg",
  "lang": "null",
  "length": 844773,
  "mime": "image/jpeg",
  "object_id": "posts/1079556",
  "path": "/Users/komal/Media/public/lm3Ty10l8XE99glBTfmANbhe1Lo3AVyTNJPLRqI3.jpg",
  "pos": 1000,
  "url": "http://localhost:8080/lm3Ty10l8XE99glBTfmANbhe1Lo3AVyTNJPLRqI3.jpg"
}
```

Frontend

AngularJS

AngularJS is a JavaScript-based open-source front-end web framework predominantly maintained by Google and by a network of people and enterprises to address a significant number of the difficulties experienced in creating single-page applications. It is structural framework that aids in making web applications to become very dynamic. It lets you extend the syntax of the simple HTML template to express the application’s components in a concise manner.

NodeJS

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

ExpressJS

Express.js is a framework used for Node and it is most commonly used as a web application for NodeJS. Express is just a module framework for Node that you can use for applications that are based on server/s that will "listen" for any input/connection requests from clients.

ArangoJS

This is ArangoDB driver that is available in npm used for connecting the ArangoDB to the Node server.

1. Install Node in your system. This includes npm which can be used to install the ArangoJS driver.

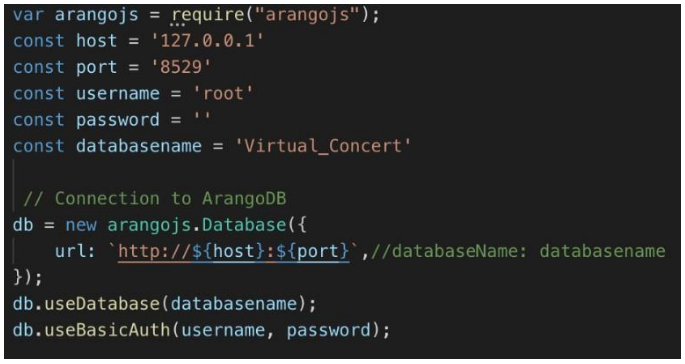
2. Create a folder for the Web Application and initialize with npm. This creates the package.json file that can be used for maintaining all installed dependencies in one place.

3. Install ArangoJS, Express

4. Create a file called app.js that serves the application by using node on the local server by mentioning the port.

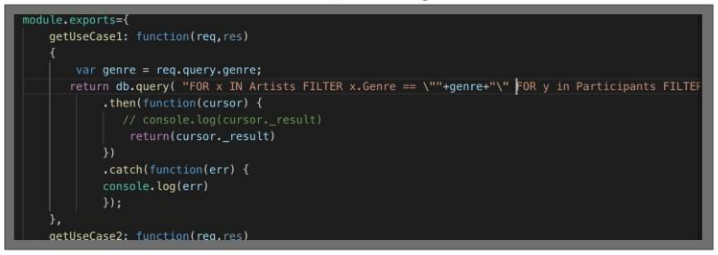
5. Define the routes namely the end points in routes.js that can be used for accessing the data services. This is where we import the express module in order to create the router.get or router.post functions.

6. All the related Data services are defined in the file called DataServices.js. Here we include the arangojs driver to ensure appropriate connection with ArangoDB. It is required to mention the host, port, username, password and the required database name to establish the connection. AQL queries can be used in the format db.query(“ ”) for querying the required information from the database.



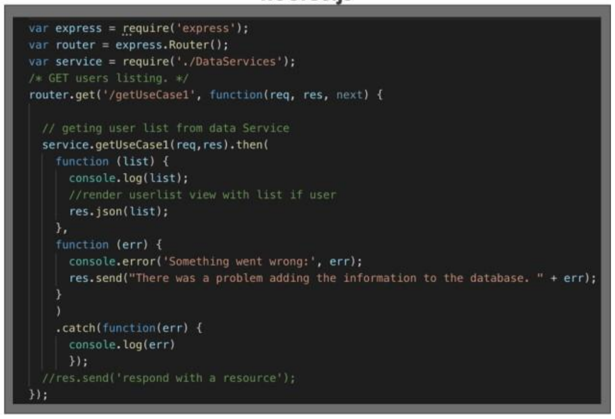
```
var arangojs = require("arangojs");
const host = '127.0.0.1'
const port = '8529'
const username = 'root'
const password = ''
const databasename = 'Virtual_Concert'

 // Connection to ArangoDB
db = new arangojs.Database({
    url: `http://${host}:${port}`,//databaseName: databasename
});
db.useDatabase(databasename);
db.useBasicAuth(username, password);
```

Data_Service.js



```
module.exports={
    getUseCase1: function(req,res)
    {
        var genre = req.query.genre;
       return db.query( "FOR x IN Artists FILTER x.Genre == \""+genre+"\" FOR y in Participants FILTER
            .then(function(cursor) {
              // console.log(cursor._result)
               return(cursor._result)
            })
            .catch(function(err) {
            console.log(err)
            });
    },
    getUseCase2: function(req,res)
```

Routes.js



```
var express = require('express');
var router = express.Router();
var service = require('./DataServices');
/* GET users listing. */
router.get('/getUseCase1', function(req, res, next) {

  // geting user list from data Service
  service.getUseCase1(req,res).then(
    function (list) {
      console.log(list);
      //render userlist view with list if user
      res.json(list);
    },
    function (err) {
      console.error('Something went wrong:', err);
      res.send("There was a problem adding the information to the database. " + err);
    }
    )
    .catch(function(err) {
      console.log(err)
      });
  //res.send('respond with a resource');
});
```

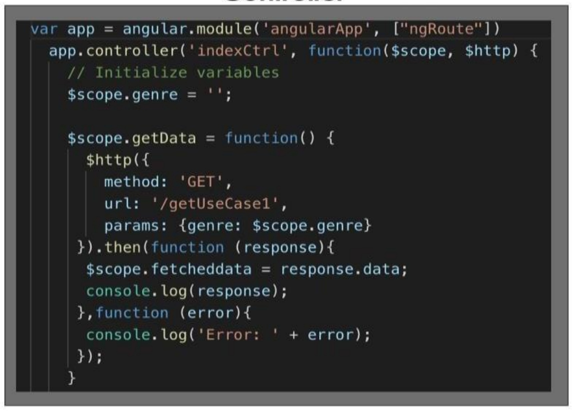
7. Once the backend is up and running with the required APIs, create a public folder which serves as the Views folder and houses all the required views to run the front end. Create an index.html that

is the starting point of the angular app. Include the cdn to the angular.js and the angular-route.js files to extend the app to a Single Page Application (SPA).

8. Create a controller for the index.html view and using the ng-route dependency it is possible to use the ng-view for dynamically changing the views on redirection.

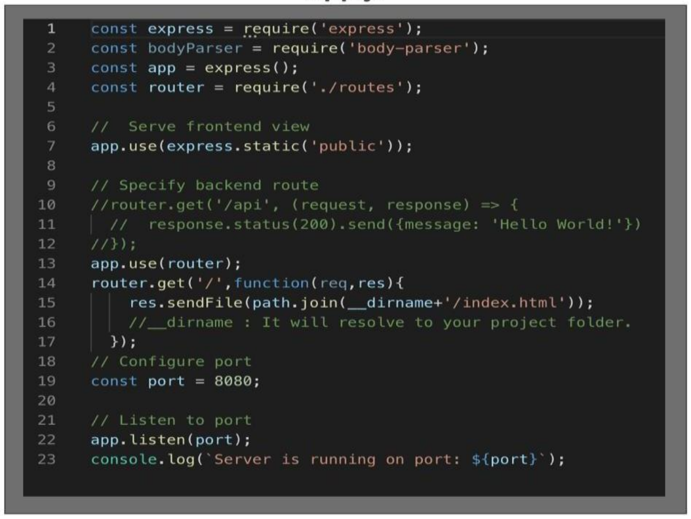
9. Inject the $http dependency so that it is possible to access the HTTP REST API services. Using this any required parameters can be passed via query string to access the required data.

Controller



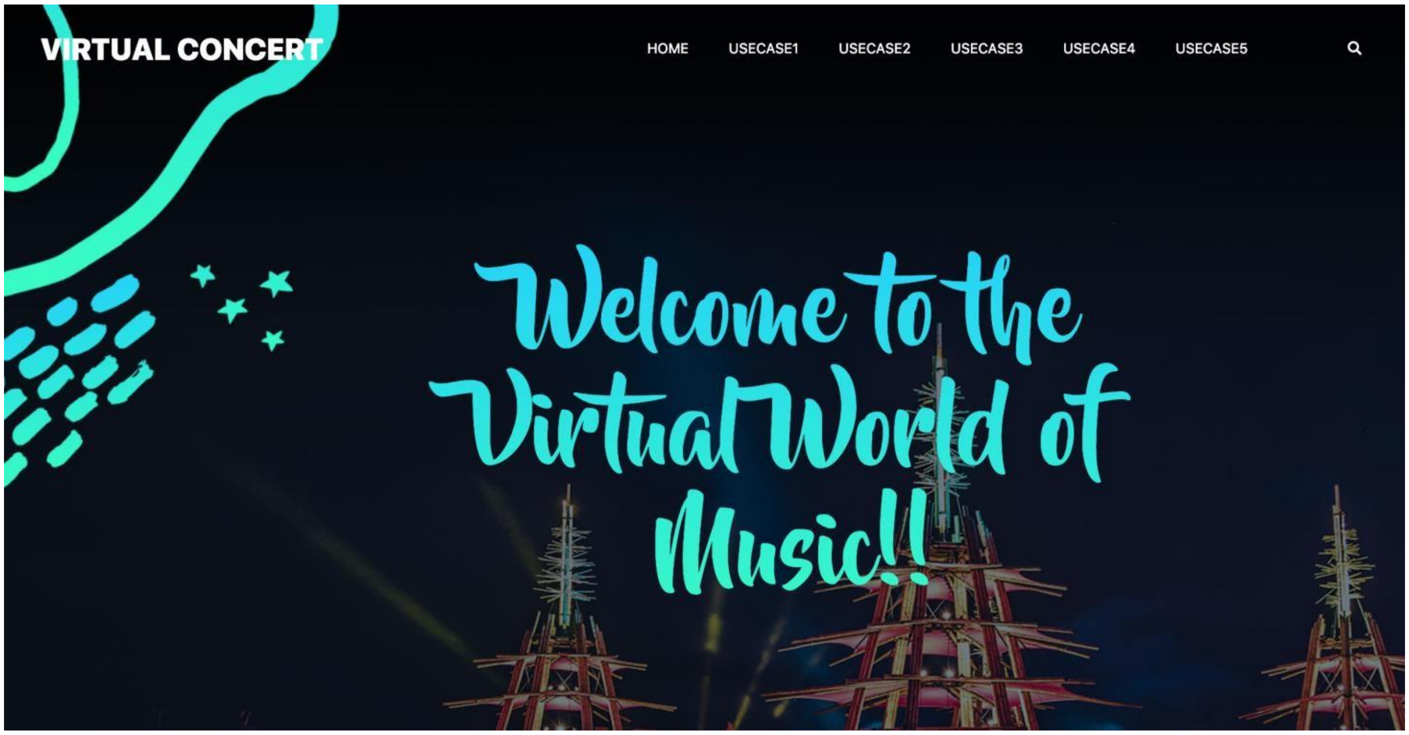
```
var app = angular.module('angularApp', ["ngRoute"])
  app.controller('indexCtrl', function($scope, $http) {
    // Initialize variables
    $scope.genre = '';

    $scope.getData = function() {
      $http({
        method: 'GET',
        url: '/getUseCase1',
        params: {genre: $scope.genre}
     }).then(function (response){
      $scope.fetcheddata = response.data;
      console.log(response);
     },function (error){
      console.log('Error: ' + error);
     });
    }
```

App.js



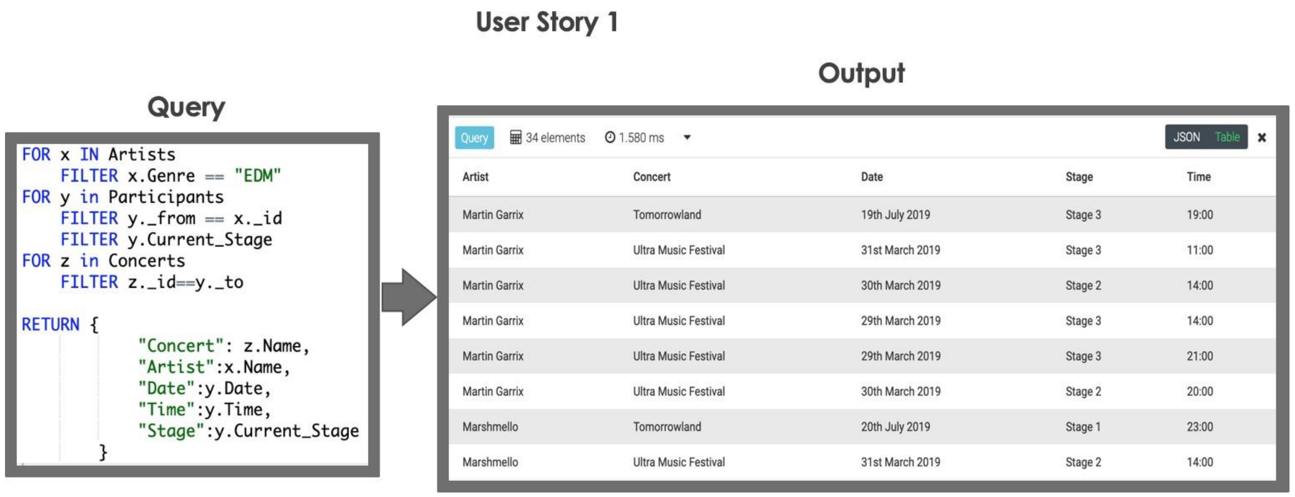
```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const router = require('./routes');

//  Serve frontend view
app.use(express.static('public'));

// Specify backend route
//router.get('/api', (request, response) => {
  //  response.status(200).send({message: 'Hello World!'})
//});
app.use(router);
router.get('/',function(req,res){
    res.sendFile(path.join(__dirname+'/index.html'));
    //__dirname : It will resolve to your project folder.
  });
// Configure port
const port = 8080;

// Listen to port
app.listen(port);
console.log(`Server is running on port: ${port}`);
```

10. Launch the app with the command: node index.js. The server should be up and running with a successful connection to ArangoDB.



7 Queries

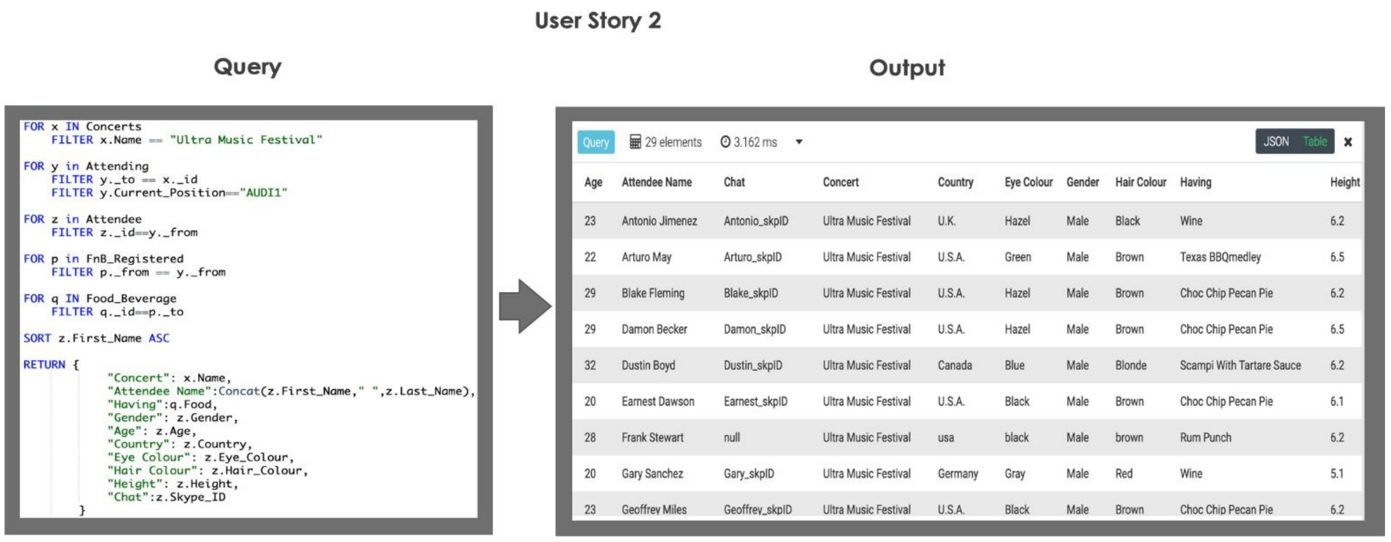
Query 1: Name all the artists and concerts where EDM genre is being played with date, time and their stage detail.



```
FOR x IN Artists
    FILTER x.Genre == "EDM"
FOR y in Participants
    FILTER y._from == x._id
    FILTER y.Current_Stage
FOR z in Concerts
    FILTER z._id==y._to

RETURN {
            "Concert": z.Name,
            "Artist":x.Name,
            "Date":y.Date,
            "Time":y.Time,
            "Stage":y.Current_Stage
        }
```

Artist	Concert	Date	Stage	Time
Martin Garrix	Tomorrowland	19th July 2019	Stage 3	19:00
Martin Garrix	Ultra Music Festival	31st March 2019	Stage 3	11:00
Martin Garrix	Ultra Music Festival	30th March 2019	Stage 2	14:00
Martin Garrix	Ultra Music Festival	29th March 2019	Stage 3	14:00
Martin Garrix	Ultra Music Festival	29th March 2019	Stage 3	21:00
Martin Garrix	Ultra Music Festival	30th March 2019	Stage 2	20:00
Marshmello	Tomorrowland	20th July 2019	Stage 1	23:00
Marshmello	Ultra Music Festival	31st March 2019	Stage 2	14:00

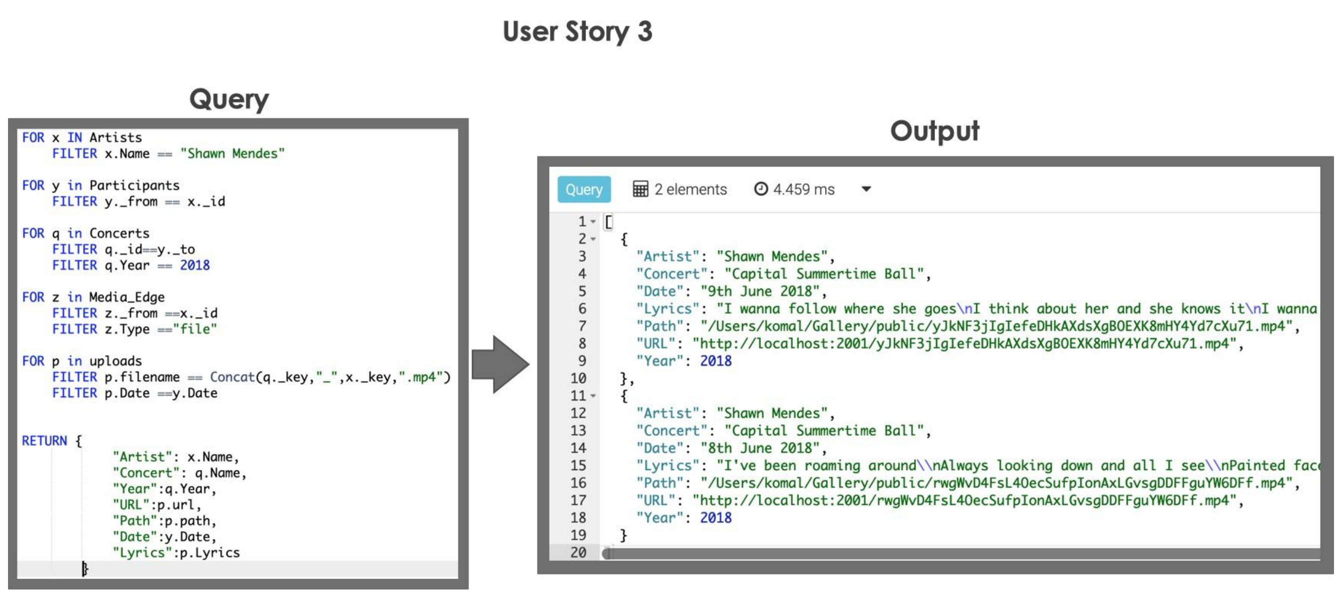
Query 2: We want list of Attendees who are participating in specific concert and specific Audi and basic information of other attendee and option to chat with them on skype id, and what he is having currently.



```
FOR x IN Concerts
    FILTER x.Name == "Ultra Music Festival"

FOR y in Attending
    FILTER y._to == x._id
    FILTER y.Current_Position=="AUDI1"

FOR z in Attendee
    FILTER z._id==y._from

FOR p in FnB_Registered
    FILTER p._from == y._from

FOR q IN Food_Beverage
    FILTER q._id==p._to

SORT z.First_Name ASC

RETURN {
            "Concert": x.Name,
            "Attendee Name":Concat(z.First_Name," ",z.Last_Name),
            "Having":q.Food,
            "Gender": z.Gender,
            "Age": z.Age,
            "Country": z.Country,
            "Eye Colour": z.Eye_Colour,
            "Hair Colour": z.Hair_Colour,
            "Height": z.Height,
            "Chat":z.Skype_ID
        }
```

Age	Attendee Name	Chat	Concert	Country	Eye Colour	Gender	Hair Colour	Having	Height
23	Antonio Jimenez	Antonio_skpID	Ultra Music Festival	U.K.	Hazel	Male	Black	Wine	6.2
22	Arturo May	Arturo_skpID	Ultra Music Festival	U.S.A.	Green	Male	Brown	Texas BBQmedley	6.5
29	Blake Fleming	Blake_skpID	Ultra Music Festival	U.S.A.	Hazel	Male	Brown	Choc Chip Pecan Pie	6.2
29	Damon Becker	Damon_skpID	Ultra Music Festival	U.S.A.	Hazel	Male	Brown	Choc Chip Pecan Pie	6.5
32	Dustin Boyd	Dustin_skpID	Ultra Music Festival	Canada	Blue	Male	Blonde	Scampi With Tartare Sauce	6.2
20	Earnest Dawson	Earnest_skpID	Ultra Music Festival	U.S.A.	Black	Male	Brown	Choc Chip Pecan Pie	6.1
28	Frank Stewart	null	Ultra Music Festival	usa	black	Male	brown	Rum Punch	6.2
20	Gary Sanchez	Gary_skpID	Ultra Music Festival	Germany	Gray	Male	Red	Wine	5.1
23	Geoffrey Miles	Geoffrey_skpID	Ultra Music Festival	U.S.A.	Black	Male	Brown	Choc Chip Pecan Pie	6.2

Query 3: We want the list of video songs played by the favorite Artists with the lyrics, Concert Name, date and year



User Story 3

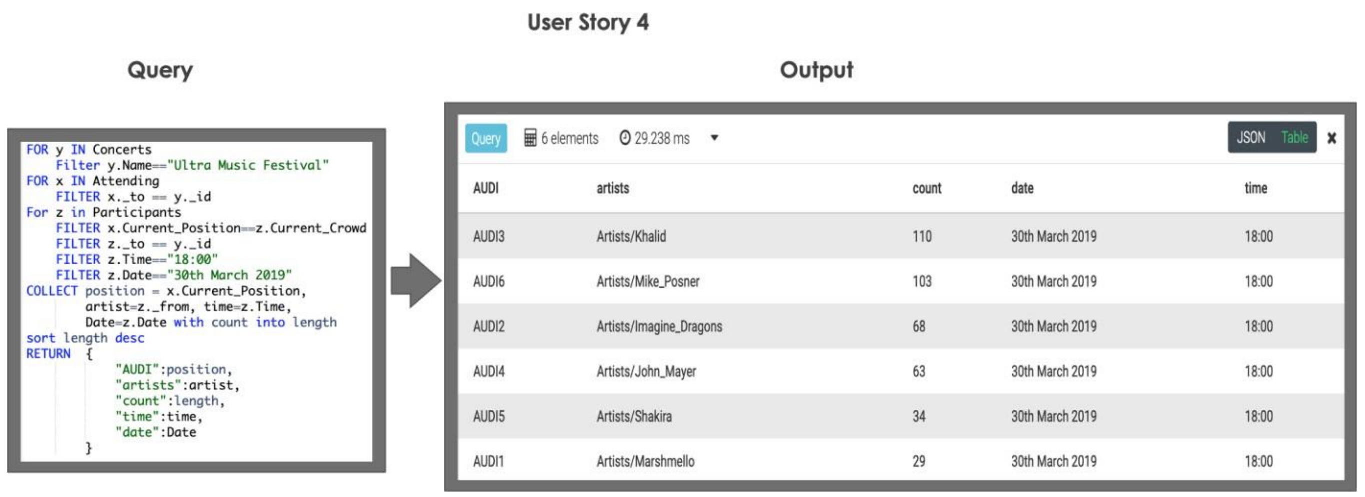
Query

```
FOR x IN Artists
    FILTER x.Name == "Shawn Mendes"

FOR y in Participants
    FILTER y._from == x._id

FOR q in Concerts
    FILTER q._id==y._to
    FILTER q.Year == 2018

FOR z in Media_Edge
    FILTER z._from ==x._id
    FILTER z.Type =="file"

FOR p in uploads
    FILTER p.filename == Concat(q._key,"_",x._key,".mp4")
    FILTER p.Date ==y.Date


RETURN {
            "Artist": x.Name,
            "Concert": q.Name,
            "Year":q.Year,
            "URL":p.url,
            "Path":p.path,
            "Date":y.Date,
            "Lyrics":p.Lyrics
        }
```

Output

```
[
  {
    "Artist": "Shawn Mendes",
    "Concert": "Capital Summertime Ball",
    "Date": "9th June 2018",
    "Lyrics": "I wanna follow where she goes\nI think about her and she knows it\nI wanna
    "Path": "/Users/komal/Gallery/public/yJkNF3jIgIefeDHkAXdsXgBOEXK8mHY4Yd7cXu71.mp4",
    "URL": "http://localhost:2001/yJkNF3jIgIefeDHkAXdsXgBOEXK8mHY4Yd7cXu71.mp4",
    "Year": 2018
  },
  {
    "Artist": "Shawn Mendes",
    "Concert": "Capital Summertime Ball",
    "Date": "8th June 2018",
    "Lyrics": "I've been roaming around\\nAlways looking down and all I see\\nPainted face
    "Path": "/Users/komal/Gallery/public/rwgWvD4FsL40ecSufpIonAxLGvsgDDFFguYW6DFf.mp4",
    "URL": "http://localhost:2001/rwgWvD4FsL40ecSufpIonAxLGvsgDDFFguYW6DFf.mp4",
    "Year": 2018
  }
```

Query 4: We want to get the recommendation for different Audi details with their capacity with time & dates so that attendee switch to another AUDI based on the number of Attendees.



User Story 4

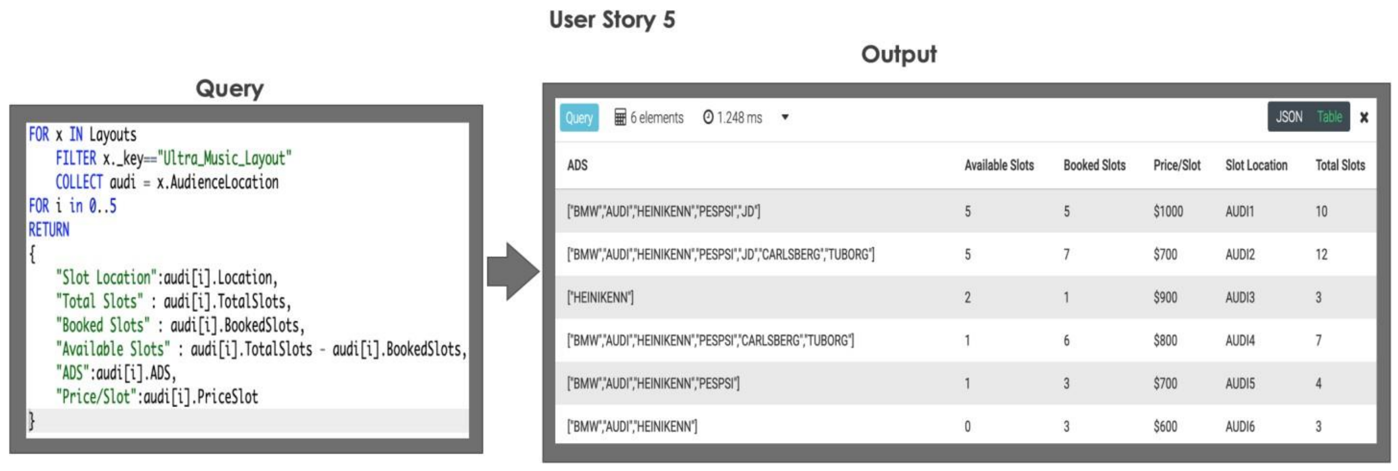
Query

```
FOR y IN Concerts
    Filter y.Name=="Ultra Music Festival"
FOR x IN Attending
    FILTER x._to == y._id
For z in Participants
    FILTER x.Current_Position==z.Current_Crowd
    FILTER z._to == y._id
    FILTER z.Time=="18:00"
    FILTER z.Date=="30th March 2019"
COLLECT position = x.Current_Position,
        artist=z._from, time=z.Time,
        Date=z.Date with count into length
sort length desc
RETURN  {
            "AUDI":position,
            "artists":artist,
            "count":length,
            "time":time,
            "date":Date
        }
```

Output

AUDI	artists	count	date	time
AUDI3	Artists/Khalid	110	30th March 2019	18:00
AUDI6	Artists/Mike_Posner	103	30th March 2019	18:00
AUDI2	Artists/Imagine_Dragons	68	30th March 2019	18:00
AUDI4	Artists/John_Mayer	63	30th March 2019	18:00
AUDI5	Artists/Shakira	34	30th March 2019	18:00
AUDI1	Artists/Marshmello	29	30th March 2019	18:00

Query 5: What are the available slots in terms of slot location, booked slots, total lots and respective price.



User Story 5

Query

```
FOR x IN Layouts
    FILTER x._key=="Ultra_Music_Layout"
    COLLECT audi = x.AudienceLocation
FOR i in 0..5
RETURN
{
    "Slot Location":audi[i].Location,
    "Total Slots" : audi[i].TotalSlots,
    "Booked Slots" : audi[i].BookedSlots,
    "Available Slots" : audi[i].TotalSlots - audi[i].BookedSlots,
    "ADS":audi[i].ADS,
    "Price/Slot":audi[i].PriceSlot
}
```

Output

ADS	Available Slots	Booked Slots	Price/Slot	Slot Location	Total Slots
["BMW","AUDI","HEINIKENN","PESPSI","JD"]	5	5	$1000	AUDI1	10
["BMW","AUDI","HEINIKENN","PESPSI","JD","CARLSBERG","TUBORG"]	5	7	$700	AUDI2	12
["HEINIKENN"]	2	1	$900	AUDI3	3
["BMW","AUDI","HEINIKENN","PESPSI","CARLSBERG","TUBORG"]	1	6	$800	AUDI4	7
["BMW","AUDI","HEINIKENN","PESPSI"]	1	3	$700	AUDI5	4
["BMW","AUDI","HEINIKENN"]	0	3	$600	AUDI6	3

8 Evaluation

1. **User Story 2**

Based on the current data model we are able to have and interact to all the attendee as we are getting whoever present in the specific auditorium but if we have to focus to get the fellow attendee info who are very close to other attendee rather than having all attendee information.

In order achieve this specific feature:

- We need to mention the longitude and latitude for each of the attendee and restrict the query based on the current position
- Attendee collection will have additional feature latitude and longitude for each attendee.
- We need to specify the distance to cover within query to get nearest attendee information.

When we want additional feature to provide attendee so that he could have option to interact with fellow attendee then:

This feature can be achieved by having an option of interacting with the other attendee with given login id which he has to put to enter in the application. With the given id he will have option of having chat box which will provide text message and call facility within the range of specific distance in specific auditorium.

2. **User Story 3**

In this User Story, we have demonstrated how to retrieve data related to performance of the artists for two different days in the same concert. For further enhancement of this query we can also include the time of the performance of different songs on the same day to categorize the video accordingly to retrieve the part of the video with respect to the name of the song or the time of the performance.

This can be resolved in few ways:

• The time related to the performance of each song can be listen in the Participants edge collection which has the date of his performance. By having these details, we can forward the playback of the video to the required song.

• Incorporating video annotations on the video of the performance can also help in achieving the required result.

• Meta data of the upload must include the song name and time attributes.

3. Use Case 5

Currently in this Use Case there is no specific time mentioned for the duration for which the Advertisement Slots are going to be available to the advertisers in the Concert. In order to achieve this, we need to create another collection termed as Advertisers which would store all the details of the advertisers who have currently booked slots in the concerts and link them to the <ConcertName_Layout> collection via an edge collection that links all the advertisers for a specific concert. The layout must also include the time duration and date specified for each slot along with the price. This would give a better picture for the sponsors.

9 Bibliography

https://www.arangodb.com/wp-content/uploads/2018/08/Switching-From-Relational-Databases-To-ArangoDB.pdf?hsCtaTracking=4cc854f1-8d25-49c3-a689-481c085e2169%7Ca97132fd-e16e-4a64-bf6e-7e2a041a92ef

https://www.arangodb.com/wp-content/uploads/2018/08/ArangoDB-White-Paper-What-is-a-multi-model-database-and-why-use-it.pdf?hsCtaTracking=964a2732-53d1-477e-93ed-0e7430c8d1bf%7C7ff1d46f-2bc6-439e-8e69-98b650993860

https://docs.arangodb.com/devel/Manual/Foxx/Guides/Files.html

https://www.arangodb.com/wp-content/uploads/2018/08/Switching-From-Relational-Databases-To-ArangoDB.pdf?hsCtaTracking=4cc854f1-8d25-49c3-a689-481c085e2169%7Ca97132fd-e16e-4a64-bf6e-7e2a041a92ef

https://foxxy.ovh

https://www.arangodb.com/wp-content/uploads/2018/08/Switching-From-Relational-Databases-To-ArangoDB.pdf?hsCtaTracking=4cc854f1-8d25-49c3-a689-481c085e2169%7Ca97132fd-e16e-4a64-bf6e-7e2a041a92ef

https://www.arangodb.com/wp-content/uploads/2018/08/Switching-From-Relational-Databases-To-ArangoDB.pdf?hsCtaTracking=4cc854f1-8d25-49c3-a689-481c085e2169%7Ca97132fd-e16e-4a64-bf6e-7e2a041a92ef

https://docs.arangodb.com/3.1/Cookbook/Graph/ExampleActorsAndMovies.html

https://en.wikipedia.org/wiki/ArangoDB

https://dzone.com/articles/arangodb-a-multi-model-nosql-database-and-how-to-u

https://cambridge-intelligence.com/powering-interactive-graph-applications-arangodb/

https://www.datanami.com/2018/03/28/arangodb-reaping-the-fruits-of-its-multi-modal-labor/

https://www.researchgate.net/publication/315564326_Movie_Management_using_ArangoDB_-_The_multi-purpose_NoSQL_database