Music Festival

Before reading this document, please read the overview to better understand what is required of this module. In a nutshell, you will need to develop and prototype, independent of the existing system of JiBaBoom, with the following functionality:

1. Basic/Advance Data Viewer – This is to view the data stored in the database, it should allow filtering and pagination
2. Basic/Advance Result Viewer – This is to view the computation result, it should allow admins to enter certain computation parameters.
3. Basic/Advance Insert API – This API will be how JiBaBoom’s existing system will insert data into the database through your Backend.
4. Basic/Advance Result API – This API will be how JiBaBoom’s existing system will get the computation result through your backend.

Table of Contents

[Problem Explanation](#_3nfe4ozfqhqy) **2**

[Basic](#_30j0zll) 2

[Problem](#_1fob9te) 2

[Data](#_3znysh7) 2

[Example Data](#_tyjcwt) 2

[Example Input](#_1t3h5sf) 2

[Example Output](#_4d34og8) 3

[Explanation](#_2s8eyo1) 3

[Advance](#_17dp8vu) 3

[Problem](#_3rdcrjn) 3

[Data](#_26in1rg) 3

[Example Data](#_35nkun2) 3

[Example Input](#_44sinio) 4

[Example Output](#_2jxsxqh) 4

[Explanation](#_z337ya) 4

[Visual Representation](#_3j2qqm3) 4

[Algorithm](#_dkr2wgp9eh2p) 5

[Basic](#_v9ch8crbl8vj) 5

[Advance](#_86k9vp5x5hqz) 5

[Frontend Requirement](#_mlufog35qmrr) 5

[Basic](#_2xcytpi) 5

[Data Viewer](#_1ci93xb) 5

[Result Viewer](#_3whwml4) 5

[Advance](#_2bn6wsx) 5

[Data Viewer](#_qsh70q) 5

[Result Viewer](#_3as4poj) 5

[Backend Requirement](#_1pxezwc) **5**

[Common Data Type](#_49x2ik5) 5

[Basic](#_2p2csry) 6

[Insert API Schema](#_147n2zr) 6

[Result API Schema](#_3o7alnk) 6

[Advance](#_23ckvvd) 7

[Insert API Schema](#_ihv636) 7

[Result API Schema](#_32hioqz) 8

# Problem Explanation

At a music festival there will be multiple stages where multiple bands play concurrently. We want to suggest to visitors, ways to best optimize their experience.

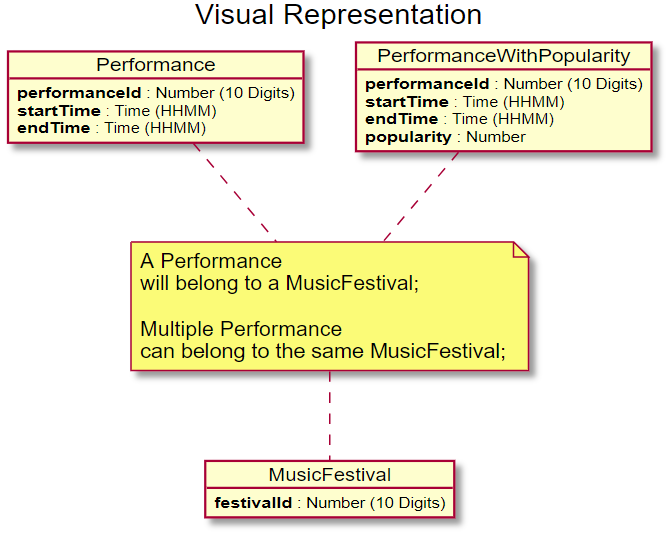
## Basic

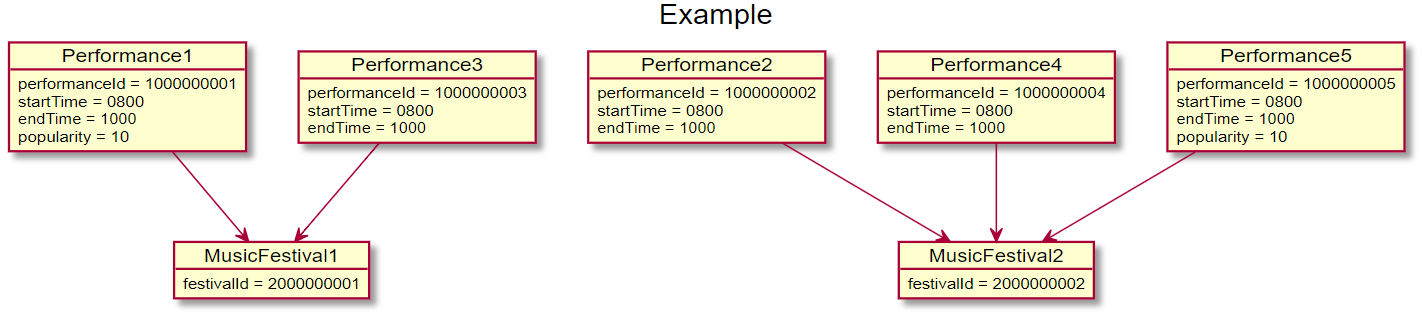
|  |  |
| --- | --- |
| Problem | Generate a schedule for visitors to follow such that they may watch the most number of performances. |
| Data (Given for Input API) | |  |  |  | | --- | --- | --- | | **Attribute** | **Data Type** | **Example** | | **performanceId** | 10 digit number | 1234567890 | | **festivalId** | 10 digit number | 1234567890 | | **startTime** | Time (HHMM) | 2359 | | **endTime** | Time (HHMM) | 2359 | |
| Example Data | |  |  |  |  | | --- | --- | --- | --- | | **Lectures** | | | | | **Attributes** | **Performance 1** | **Performance 2** | **Performance 3** | | **performanceId** | 1111111111 | 2222222222 | 3333333333 | | **festivalId** | 3333333333 | 3333333333 | 3333333333 | | **startTime** | 1000 | 1100 | 1200 | | **endTime** | 1200 | 1300 | 1400 |   Visually:   |  |  |  |  | | --- | --- | --- | --- | | 1000 - 1100 | 1100 - 1200 | 1200 - 1300 | 1300 - 1400 | | Performance 1 | |  |  | |  | Performance 2 | |  | |  |  | Performance 3 | | |
| Example Input | |  |  | | --- | --- | | **Attribute** | **Value** | | **festivalId** | 3333333333 | |
| Example Output | |  |  |  |  | | --- | --- | --- | --- | | # | Performance | From Time | To Time | | 1 | Performance 1 | 1000 | 1200 | | 2 | Performance 3 | 1200 | 1400 |   *Note: We do not allow partial participation of a performance. E.g. saying to attend performance 1 from 1000 to 1100, followed by performance 2 from 1100 to 1300 and then performance 3 from 1300 to 1400 is not a valid solution.* |
| Explanation | Visually, we can clearly see that if we include Performance 2, we won’t be able to make it for performance 1 and 3. |

## Advance

|  |  |
| --- | --- |
| Problem | Sometimes participants want to watch as many popular performances as possible (in contrast to simply watching any performance). We want to create a schedule such that the sum of popularity score is maximized. |
| Data (Given for Input API) | |  |  |  | | --- | --- | --- | | **Attribute** | **Data Type** | **Example** | | **performanceId** | 10 digit number | 1234567890 | | **festivalId** | 10 digit number | 1234567890 | | **startTime** | Time (HHMM) | 2359 | | **endTime** | Time (HHMM) | 2359 | | **popularity** | Number | 10000 |   *Note: The addition of popularity* |
| Example Data | |  |  |  |  | | --- | --- | --- | --- | | **Lectures** | | | | | **Attributes** | **Performance 1** | **Performance 2** | **Performance 3** | | **performanceId** | 1111111111 | 2222222222 | 3333333333 | | **festivalId** | 3333333333 | 3333333333 | 3333333333 | | **startTime** | 1000 | 1100 | 1200 | | **endTime** | 1200 | 1300 | 1400 | | **popularity** | 1 | 10000 | 1 |   Visually:   |  |  |  |  | | --- | --- | --- | --- | | 1000 - 1100 | 1100 - 1200 | 1200 - 1300 | 1300 - 1400 | | Performance 1 (1) | |  |  | |  | Performance 2 (10000) | |  | |  |  | Performance 3 (1) | | |
| Example Input | |  |  | | --- | --- | | **Attribute** | **Value** | | **festivalId** | 3333333333 | |
| Example Output | |  |  |  |  |  | | --- | --- | --- | --- | --- | | # | Performance | From Time | To Time | Popularity | | 1 | Performance 2 | 1100 | 1300 | 10000 | | Total Popularity | | | | 10000 |   *Note: We do not allow partial participation of a performance. E.g. saying to attend performance 1 from 1000 to 1100, followed by performance 2 from 1100 to 1300 and then performance 3 from 1300 to 1400 is not a valid solution.* |
| Explanation | Visually, we can clearly see that if we include Performance 1 & 3, we will only get a total popularity of 2. Which is clearly not the maximum. |

# Visual Representation





# Algorithm

## Basic

Name of problem: **Activity Selection**

1. Correctly select the set of performances for computation
2. Sort the performances by increasing order of their finishing time.
3. Maintain a list of selected performance
4. Iterate through each of the sorted performance
   1. If it doesn't clash with any selected performance, add it to the list
   2. Otherwise skip that performance.

## Advance

Name of problem: **Weighted Activity Selection**

*Hint: It is sufficient to implement a linear search instead of a binary search.*

# Frontend Requirement

## Basic

|  |  |
| --- | --- |
| Data Viewer | Filter Performance By:   1. festivalId 2. startTime (Performances that has startTime greater or equal(>=) to the filter start time) |
| Result Viewer | Computation Input:   1. festivalId |

## Advance

|  |  |  |
| --- | --- | --- |
| Data Viewer | Filter Performance By:   1. festivalId 2. startTime (Performances that has startTime greater or equal(>=) to the filter start time) 3. endTime (Performances that has endTime smaller (<) than the filter end time) |  |
| Result Viewer | Computation Input:   1. festivalId |  |

# Backend Requirement

## Common Data Type

|  |  |  |
| --- | --- | --- |
| Name | Format | Example |
| TIME | String (HHMM) | “2359” |
| DATE | String (YYYY/MM/DD) | “2020/02/13” |
| IDENTIFIER | Number (10 Digits) | 1234567890 |

## Basic

|  |  |
| --- | --- |
| Insert API Schema | POST /basic/insert  Request Body  {  "data": [  {  "festivalId": IDENTIFIER,  "performanceId": IDENTIFIER,  "startTime": TIME,  "endTime": TIME,  }  ]  }  Response Body  {  "result": "success"  }  Error Response  {  "error": string,  "code": number  }  Sample Request  {  "data": [  {  "festivalId": 1234567890,  "performanceId": 1234567890,  "startTime": 1000,  "endTime": 1030,  },  {  "festivalId": 1234567891,  "performanceId": 1234567891,  "startTime": 1030,  "endTime": 1100,  },  ]  }  Sample Error  {  "error": "Duplicate Entry",  "code": 400,  } |
| Result API Schema | GET /basic/result  Query String  {  "festivalId": IDENTIFIER  }  Response Body  {  "result": [  [  {  "performanceId": IDENTIFIER,  "startTime": TIME,  "endTime": TIME  }  ]  ]  }  Error Response  {  "error": string,  "code": number  }  Sample Request  /basic/result?festivalId=1234567890  Sample Response  {  "result": [  [  {  "performanceId": 1234567890,  "startTime": 1000,  "endTime": 1030  },  {  " performanceId ": 1234567891,  "startTime": 1030,  "endTime": 1100  }  ]  ]  }  Sample Error  {  "error": "Server Error",  "code": 500,  } |

## Advance

|  |  |  |
| --- | --- | --- |
| Insert API Schema | POST /basic/insert  Request Body  {  "data": [  {  "festivalId": IDENTIFIER,  "performanceId": IDENTIFIER,  "startTime": TIME,  "endTime": TIME,  "popularity": Number,  }  ]  }  Response Body  {  "result": "success"  }  Error Response  {  "error": string,  "code": number  }  Sample Request  {  "data": [  {  "festivalId": 1234567890,  "performanceId": 1234567890,  "startTime": 1000,  "endTime": 1030,  "popularity": 10000  },  {  "festivalId": 1234567891,  "performanceId": 1234567891,  "startTime": 1030,  "endTime": 1100,  "popularity": 1000  },  ]  }  Sample Error  {  "error": "Duplicate Entry",  "code": 400,  } |  |
| Result API Schema | GET /basic/result  Query String  {  "festivalId": IDENTIFIER  }  Response Body  {  "result": [  [  {  "performanceId": IDENTIFIER,  "startTime": TIME,  "endTime": TIME,  "popularity": Number  }  ]  ]  }  Error Response  {  "error": string,  "code": number  }  Sample Request  /basic/result?festivalId=1234567890  Sample Response  {  "result": [  [  {  "performanceId": 1234567890,  "startTime": 1000,  "endTime": 1030,  "popularity": 10000  },  {  " performanceId ": 1234567891,  "startTime": 1030,  "endTime": 1100,  "popularity": 1000  }  ]  ]  }  Sample Error  {  "error": "Server Error",  "code": 500,  } |  |