API Documentation

1. Mastery-based Coaching

<table>
<tr><td colspan="2">Type: GET</td><td>Description:
To retrieve a coach</td></tr>
<tr><td colspan="3">URL: /jsonapi/coach</td></tr>
<tr><td>Parameters: -</td><td colspan="2">Payload : -</td></tr>
<tr><td colspan="2">Response:
{"coachesData":
[{
"coachID":integer,
"name":String
"image":String,
"description": String,

"audiofile":{
"greeting": String,
"welcomeback": String,
"areyouthere":String,
"dontgiveup":String,
"correctanswer": String,
"tryother": String,
"faster": String,
"lessattempts": String
},

"audiotext":{"greeting": String,
"welcomeback": String,
"areyouthere":String,
"dontgiveup":String,
"correctanswer": String,
"tryother": String,
"faster": String,
"lessattempts": String
},</td><td>coachData
Array of coach objects
coachID
Unique id for the coach
name
Name of the coach
image
Default image to show when selecting coach
description
Description of the coach

audiofile
the specific attribute of the audifile such as “greeting” or “welcomeback” acts as a key that stores in the value of the audio_file location for that specific key attribute.

audiotext
the specific attribute of the audiotext such as “greeting” or “welcomeback” acts as a key that stores in the string of what is being said to it’s audiofile counterpart.</td></tr>
</table>

"pictures": {"greeting": String, "welcomeback": String, "areyouthere":String, "dontgiveup":String, "correctanswer": String, "tryother": String, "faster": String, "lessattempts": String } }]}	**Pictures** the specific attribute of the pictures such as “greeting” or “welcomeback” acts as a key that stores in the location of the image to be shown when it plays the audio of its audiofile counterpart.
Comments:	

Sample Output of GET: /jsonapi/coach

{"coachesData":
[{"coachID":1,"name":"Shannon","image":"img/mbcoach/Shannon/Shannon.jpg","description":"Encourager that wants you to be ready to code with your friends",

"audiofile":{"greeting":"audio/Shannon/greeting.mp3","welcomeback":"audio/Shannon/welcomeback.mp3","areyouthere":"audio/Shannon/areyouthere.mp3","dontgiveup":"audio/Shannon/dontgiveup.mp3","correctanswer":"audio/Shannon/correctanswer.mp3","tryother":"audio/Shannon/tryother.mp3","faster":"audio/Shannon/faster.mp3","lessattempts":"audio/Shannon/lessattempts.mp3"},

"audiotext":{"greeting":"Hi I am Shannon, I am here to help you practice and resolve some of the problems you have seen before.By resolving this problems, you'll be a little better prepared the next time you get together with your friends to do some coding.","welcomeback":"Welcome back! You were on a roll the last time. If you keep coding like this every day, your friends are going to come to you for help! Now, let's start on this question!","areyouthere":"Hey – are you there? I hope you are not giving up! Let's finish these problems together! ","dontgiveup":"Hmm…there seems to be an error. Check the compiler. It will help you solve it quicker.","correctanswer":"Alright! That was a good one! Let's move onto the next one!","tryother":"You are on a roll! Here, try another question!","faster":"This question shouldn't take you so long – I believe in you!","lessattempts":"Now here's a problem that I think you can do in fewer attempts. "},

"pictures":{"greeting":"img/mbcoach/Shannon/Shannon.jpg","welcomeback":"img/mbcoach/Shannon/Shannon.jpg","areyouthere":"img/mbcoach/Shannon/Shannon.jpg","dontgiveup":"img/mbcoach/Shannon/Shannon.jpg","correctanswer":"img/mbcoach/Shannon/Shannon.jpg","tryother":"img/mbcoach/Shannon/Shannon.jpg","faster":"img/mbcoach/Shannon/Shannon.jpg","lessattempts":"img/mbcoach/Shannon/Shannon.jpg"
}]

}

<table>
<tr><td colspan="2">Type: GET</td><td>Description:
To retrieve the coach status</td></tr>
<tr><td colspan="3">URL: /jsonapi/current_coaching_status</td></tr>
<tr><td colspan="2">Parameters:</td><td>Payload :</td></tr>
<tr><td colspan="2">Response:
var currentUserMasteryProgress = {
"showNewProblems":boolean,
"nextProblemID": integer,
"coach":String,
"goal": String

"next_ten":
[{"percentile_time":float, "percentile_attempts": float, "problemId": integer }],

"fromProblemSetID": integer,
"coachID": integer,

"past_result":
{"problemID": integer, "name":" String ", "goal":" String ", "percent_improvement": integer },

"pathName":" String ",
"pathID":" integer "
};</td><td rowspan="2">currentUserMasteryProgress
shows the current status of users playing Mastery
showNewProblems
determines if a new problems will occur in game.
nextProblemID
problem Id of the next problem to solve.
Coach
The last coach chosen and will be defaulted to
goal
determines if a user requires “faster” or “less attempts”
next_time
An Array that an object with the players’ 10 least performing problems. It gives the percentile of time taken, total attempts percentile for a specific problem ID.
fromProblemSetID
The problemSet that the nextProblemID is from
coachID
ID of the coach
past_result
yet to be implemented.
pathName
name of the language the user currently selects
pathID
ID of the pathName</td></tr>
<tr><td colspan="2">Comments:</td></tr>
</table>

Sample Output of GET: / jsonapi/current_coaching_status

currentUserMasteryProgress = {
"showNewProblems":true,
"nextProblemID": 52741,
"coach":"Shannon",
"goal": "faster", // or lessattempts

"next_ten":

[{"percentile_time":0.1,"percentile_attempts":0.2,"problemId":52741},
{"percentile_time":0.2,"percentile_attempts":0.1,"problemId":52472},
{"percentile_time":0.2,"percentile_attempts":0.1,"problemId":57555},
{"percentile_time":0.11,"percentile_attempts":0.0023,"problemId":52747},
{"percentile_time":0.001,"percentile_attempts":0.0023,"problemId":52748},
{"percentile_time":0.001,"percentile_attempts":0.0023,"problemId":52749},
{"percentile_time":0.001,"percentile_attempts":0.0023,"problemId":52747},
{"percentile_time":0.001,"percentile_attempts":0.0023,"problemId":52748},
{"percentile_time":0.001,"percentile_attempts":0.0023,"problemId":52749},
{"percentile_time":0.001,"percentile_attempts":0.0023,"problemId":52740}],

"fromProblemSetID":10041,
"coachID":4,
"past_result":{"problemID":10033, "name":"Expected Results", "goal":"faster", "percent_improvement":22},
"pathName":"python",
"pathID":"10030"
};

2. Purpose Driven

<table>
<tr><td colspan="2">Type: GET</td><td>Description:
To retrieve the specific video based on the video ID</td></tr>
<tr><td colspan="3">URL: /jsonapi/purposevideos</td></tr>
<tr><td colspan="2">Parameters:</td><td>Payload :</td></tr>
<tr><td colspan="2">Response:
{
"Videos":[
{
"id" : integer,
"no" : integer,
"title" : String,
"image" : String,
"thumbnail": String,
"vlink": String,
"description" : String ,
"selected": integer,
"unlocked": Boolean
}
]

}</td><td rowspan="2">Videos
Array of videos that include details such as ID, title, image and such
ID
Unique id for the video
no
Number of the video in a gallery
title
The name of the videos (i.e. “What Most School Don't Teach")
image
Previously this image file (path) was used as a placeholder
thumbnail
File path of the image used as a thumbnail for the video
vlink
The YouTube link of the video
description
Short write-up of what the video is about
question
The prompted question to the user
feedback
Record response user enters
Unlocked
Boolean to check if user had already unlocked this vid</td></tr>
<tr><td colspan="2">Comments:</td></tr>
</table>

Sample Output of GET : /jsonapi/purposevideos

{"Videos":[

{"id":01,"no":0,"title":"What Most School Don't Teach","image":"img/purposedrivenPlaceholder/PurposeDriven0.jpg","thumbnail":"img/purposedrivenPlaceholder/thumb/0.jpg","vlink":"http://www.youtube.com/watch?v=nKIu9yen5nc","description":"Learn about a new 'superpower' that isn't being taught in in 90% of US schools. Starring Bill Gates, Mark Zuckerberg, will.i.am, Chris Bosh, Jack Dorsey, Tony Hsieh, Drew Houston, Gabe Newell, Ruchi Sanghvi, Elena Silenok, Vanessa Hurst, and Hadi Partovi. D","question":"How does this video make you feel?","feedback":1,"unlocked":true},

{"id":02,"no":1,"title":"Art of Creative Coding","image":"img/purposedrivenPlaceholder/PurposeDriven1.jpg","thumbnail":"img/purposedrivenPlaceholder/thumb/1.jpg","vlink":"http://www.youtube.com/watch?v=eBV14-3LT-g","description":"Programming plays a huge role in the world that surrounds us, and though its uses are often purely functional, there is a growing community of artists who use the language of code as their medium.","question":"How does this video make you feel?","feedback":3,"unlocked":true}

{"id":900,"no":9,"title":"NEW NEWS : Computer coding for kids","image":"img/purposedrivenPlaceholder/PurposeDriven4.jpg","thumbnail":"img/purposedrivenPlaceholder/thumb/4.jpg","vlink":"http://www.youtube.com/watch?v=WGZioLhbZ6g","description":"Teaching kids how to write computer programs, by Marshall Brain marshallbrain.com/kids-programming.htm - Traduzir esta página Let's say that you have children, and you would like to help them learn computer programming at a youngish age.","question":"How does this video make you feel?","feedback":0,"unlocked":false}

]}

<table>
<tr><td colspan="2">Type: POST</td><td>Description:
To unlocked the next video and saving the feedback value of the last watched video</td></tr>
<tr><td colspan="3">URL: /jsonapi/record_purpose_video_unlock/</td></tr>
<tr><td colspan="2">Payload :
{
"feedback": integer,
"purposevideo": integer
}</td><td>Parameters :
1.feedback refers to the value of the input selected by the user upon watching the video.
2.purposevideo refers to the position of video in the arraylist that was retrieves in /jsonapi/purposevideos</td></tr>
<tr><td colspan="2">Response:
{ "result": String
}</td><td rowspan="2">Result
will just give status base on the Post. All successful POST of this API leads to this response.</td></tr>
<tr><td colspan="2">Comments:</td></tr>
</table>

Sample Payload of POST : /jsonapi/record_purpose_video_unlock/

```
$scope.userCurrentVideo = $resource("/jsonapi/record_purpose_video_unlock/");
var data = {"purposevideo": 3,"feedback":0};
var item = new $scope.userCurrentVideo(data);
item.$save(function(response) {
$scope.response = response;
})
```

Sample Output of POST : /jsonapi/record_purpose_video_unlock/

{"result":"video 3 unlocked and updated"})

3. School Map & Registration

Type: GET	**Description:** Retrieves all SG schools in the database
URL: /jsonapi/schools/SG	
Parameters:-	**Payload :-**

Response:


```
{
   "University": [
      {
         "name": "Singapore Management University",
         "schooltype": "University",
         "created": "2013-10-18T06:52:15.473820",
         "longitude": 103.849884,
         "subtype": "Local",
         "latitude": 1.2966608,
         "id": 4838709618802688
      },
      {
         "name": "Singapore University of Technology and Design",
         "schooltype": "University",
         "created": "2013-10-18T06:52:15.563320",
         "longitude": 103.78118,
         "subtype": "Local",
         "latitude": 1.300555,
         "id": 5033984601882624
      },
      {
         "name": "National University of Singapore",
         "schooltype": "University",
         "created": "2013-10-18T06:52:15.369270",
         "longitude": 103.770355,
         "subtype": "Local",
         "latitude": 1.2933539,
         "id": 5445803246092288
```


Grouped by School Type

- University
- Tertiary
- Secondary

name
School name
schoolType
School type based on level
created
Time stamp school was added to database
longitude
Longitude of the school
subtype
Subtype of school. Used primarily to differentiate between junior colleges and polytechnics at the tertiary level
latitude
Latitude of School
id
ID of the school

```
    },
    {
      "name": "SIM University",
      "schooltype": "University",
      "created": "2013-10-18T06:52:15.772720",
      "longitude": 103.77585,
      "subtype": "Local",
      "latitude": 1.328807,
      "id": 5560500347731968
    },
    {
      "name": "Singapore Institute of Technology",
      "schooltype": "University",
      "created": "2013-10-18T06:52:15.675140",
      "longitude": 103.849815,
      "subtype": "Local",
      "latitude": 1.290036,
      "id": 6324287266881536
    },
    {
      "name": "Nanyang Technical Univeristy",
      "schooltype": "University",
      "created": "2013-10-18T06:52:15.197620",
      "longitude": 103.68101,
      "subtype": "Local",
      "latitude": 1.344557,
      "id": 6555038679826432
    }
  ]
}
```


Comments:

Sample Output of GET: /jsonapi/schools/SG

{"University":[{"name":"Singapore Management University","schooltype":"University","created":"2013-10-18T06:52:15.473820","longitude":103.849884,"subtype":"Local","latitude":1.2966608,"id":4838709618802688},{"name":"Singapore University of Technology and Design","schooltype":"University","created":"2013-10-18T06:52:15.563320","longitude":103.78118,"subtype":"Local","latitude":1.300555,"id":5033984601882624},{"name":"National University of Singapore","schooltype":"University","created":"2013-10-18T06:52:15.369270","longitude":103.770355,"subtype":"Local","latitude":1.2933539,"id":5445803246092288},{"name":"SIM University","schooltype":"University","created":"2013-10-18T06:52:15.772720","longitude":103.77585,"subtype":"Local","latitude":1.328807,"id":5560500347731968},{"name":"Singapore Institute of Technology","schooltype":"University","created":"2013-10-18T06:52:15.675140","longitude":103.849815,"subtype":"Local","latitude":1.290036,"id":6324287266881536},{"name":"Nanyang Technical Univeristy","schooltype":"University","created":"2013-10-18T06:52:15.197620","longitude":103.68101,"subtype":"Local","latitude":1.344557,"id":6555038679826432}]}

<table>
<tr><td colspan="2">Type: GET</td><td>Description:
Retrieves all users who have registered their SG schools to their account</td></tr>
<tr><td colspan="3">URL: /jsonapi/school_registration</td></tr>
<tr><td colspan="2">Parameters:</td><td>Payload :</td></tr>
<tr><td colspan="2">Response: example of the response body – include the name and type
[
{
"school": 5201806154006528,
"schooltype": "Secondary",
"subtype": "",
"created": "2013-10-18T09:51:42.456680",
"player": 50001,
"year": 2010,
"id": 4527977324871680
}
]</td><td rowspan="2">Describe each of the parameter in the response body and what it’s for. Here’s a sample:

school
id for school
schooltype
Type of school based on level
subtype
Sub type of school to differentiate within levels
created
Time stamp for when entry was created
player
Player ID
year
Year first started in that school
id
ID for specific entry</td></tr>
<tr><td colspan="2">Comments: is there anything that needs to be noted?</td></tr>
</table>

Sample Output of GET: /jsonapi/school_registration


```
[
  {
    "school": 5201806154006528,
    "schooltype": "Secondary",
    "subtype": "",
    "created": "2013-10-18T09:51:42.456680",
    "player": 50001,
    "year": 2010,
    "id": 4527977324871680
  },
  {
    "school": 5201806154006528,
    "schooltype": "Secondary",
    "subtype": "",
    "created": "2013-10-18T09:51:42.456680",
    "player": 50000,
    "year": 2008,
    "id": 4527977324871680
  },
  {
    "school": 5201806154006528,
    "schooltype": "Secondary",
    "subtype": "",
    "created": "2013-10-18T09:51:42.456680",
    "player": 57754,
    "year": 2008,
    "id": 4527977324871680
  }
]
```

4. Tournament (Create & Manage)

<table>
<tr><td>Type: POST</td><td>Description:
To create new tournament or edit existing tournament</td></tr>
<tr><td colspan="2">URL:
/jsonapi/create_or_update_tournament
&
/jsonapi/create_or_update_tournament/ + tournamentID</td></tr>
<tr><td>Payload :
Create Tournament
{
"shortTitle": String,
"description”: String,
"password": String,
"status": "Closed",
"type": "Genshyft",
"details": String,
"isGroup": Boolean,
"assignMentorInTeam": Boolean,
"maxGroups": Integer,
"maxPlayersPerGroup": Integer
}

Edit Tournament
{
“tournamentID”: Integer,
"shortTitle": String,
"description”: String,
"password": String,
"status": "Closed",
"type": "Genshyft",
"details": String,
"isGroup": Boolean,
"assignMentorInTeam": Boolean,
"maxGroups": Integer,
"maxPlayersPerGroup": Integer</td><td>Parameters :
1. shortTitle – Title of the tournament.
2. description – short description of the tournament.
3. password – password for players to register for the tournament.
4. status – status of the tournament. Initialized as ‘Closed’.
5. details – additional details of the tournament
6. isGroup – Boolean value to indicate whether tournament is a group tournament.
7. assignMentorInTeam - Boolean value to indicate whether mentor assignment is done within the team for a group tournament.
8. maxGroups – maximum number of groups for a group tournament. Default value is 0 if it is not a group tournament.
9. maxPlayersPerGroup - maximum number of players per group for a group tournament. Default value is 0 if it is not a group tournament.
10. tournamentID – tournament ID of the tournament to be edited.</td></tr>
</table>

}	

Response:
Create Tournament
Error
{
“printing response here:" + JSON.stringify(response),
“printing error in response here:" + response.error
}
 	
Success
{
"Successfully Save Group tournament into DB",
“tournamentID”
}
 	
Edit tournament
Error
{
“Printing Error Here: " + response.error
}
 	
Success
{
“Save edited tournament details into DB"
}
	Result
	will just give status base on the Post. All successful POST of this API leads to this response.

Comments:	

Sample Payload of POST : /jsonapi/create_or_update_tournament

```
var data = {"shortTitle": “JavaScript Group Tournament”,
            "description": “Group Tournament Example”,
            "password": “Password”,
            "status": "Closed",
            "type": "Genshyft",
            "details": “Additional details for tournament”,
            "isGroup":  true,
            "assignMentorInTeam": true,
            "maxGroups": 10,
            "maxPlayersPerGroup": 3}
    $scope.NewGrpTournament = $resource('/jsonapi/create_or_update_tournament');
    var new_grpTournament = new $scope.NewGrpTournament(data);
    new_grpTournament.$save(function(response){
     if(response.error) {
       console.log("printing response here:" + JSON.stringify(response));
       console.log("printing error in response here:" + response.error);
     }

     console.log("Successfully Save Group tournament into DB")
     $scope.createdTournament = response;
     console.log($scope.createdTournament.id);
    })
```

Sample Output of POST : /jsonapi/ create_or_update_tournament

{"Successfully Save Group tournament into DB", 123456}

Sample Payload of POST : /jsonapi/create_or_update_tournament/ + *tournamentID*

```
var data = {“tournamentID”: 123456,
            "shortTitle": “JavaScript Group Tournament”,
            "description": “Group Tournament Example”,
            "password": “Password”,
            "status": "Closed",
            "type": "Genshyft",
            "details": “Additional details for tournament”,
            "isGroup":  true,
            "assignMentorInTeam": true,
            "maxGroups": 10,
            "maxPlayersPerGroup": 3}
    $scope.NewTournament = $resource('/jsonapi/create_or_update_tournament/'+ 123456);
    var new_tournament = new $scope.NewTournament(updatedTournament);
    new_tournament.$save(function(response){
     if(response.error) {
       console.log("Printing Error Here: " + response.error)
     }
     console.log("Save edited tournament details into DB")
    });
```

Sample Output of POST : /jsonapi/ create_or_update_tournament + *tournamentID*

```
{" Save edited tournament details into DB"}
```

<table>
<tr><td colspan="2">Type: POST</td><td>Description:
To create new rounds or edit existing rounds</td></tr>
<tr><td colspan="3">URL:
/jsonapi/add_or_update_round
&
/jsonapi/ add_or_update_round / + roundID</td></tr>
<tr><td>Payload :
Create Round
{
'timelimit': Integer * 60,
'description': String,
'problemIDs': Array of Integers,
'tournamentID': Integer
}

Edit Round
{
"roundID": Integer,
"timelimit": Integer * 60,
"problemIDs": Array of Integers,
"description": String
}</td><td colspan="2">Parameters :
1. timelimit – duration of each round in seconds
2. description – title of the round
3. problemIDs – array of problem IDs for the round
4. tournamentID – tournament ID of the tournament the round belongs to
5. roundID – round ID of the round to be edited</td></tr>
<tr><td colspan="2">Response:
Create Round
Error
{
String
}

Success
{
“Successfully Save round into DB”,
JSON.stringify($scope.round)
}

Edit Round</td><td>Result
will just give status base on the Post. All successful POST of this API leads to this response.</td></tr>
</table>

Error
{
"Printing Error Here: " + response.error
}
 	
Success
{
"Save edited round details into DB"
}	

Comments:	

Sample Payload of POST: /jsonapi/add_or_update_round

```
var data = {'timelimit': 3600,
          'description': “Fun Round”,
          'problemIDs': [1234,2345,3456,5345],
          'tournamentID': 123456
            }
    $scope.NewRound = $resource('/jsonapi/add_or_update_round');
    var new_round = new $scope.NewRound(data);
    new_round.$save(function(response){
     if(response.error) {
      console.log(response.error)
     }
     else{
      console.log("Successfully Save round into DB");
      $scope.round = response;
      console.log(JSON.stringify($scope.round))
})
```

Sample Output of POST: /jsonapi/add_or_update_round

```
{"Successfully Save round into DB",
'timelimit': 3600,  'description': “Fun Round”, 'problemIDs': [1234,2345,3456,5345], 'tournamentID': 123456,’roundID’:1111
}
```

Sample Payload of POST: /jsonapi/add_or_update_round/ + *roundID*

```
$scope.NewRound = $resource('/jsonapi/add_or_update_round/'+ 1111);
    var new_round = new $scope.NewRound(updatedRound);
    new_round.$save(function(response){
      if(response.error) {
        console.log("Printing Error Here: " + response.error)
      }
      //$scope.round = response;
      console.log("Save edited round details into DB")
    });
```

Sample Output of POST: /jsonapi/add_or_update_round + *roundID*

```
{" Save edited round details into DB "}
```

<table>
<tr><td colspan="2">Type: POST</td><td>Description:
To stop any ongoing heat for current round</td></tr>
<tr><td colspan="3">URL:
/jsonapi/stop_heat_updated</td></tr>
<tr><td colspan="2">Payload :
{
“tournamentID”: integer
“roundID”: integer
}</td><td>Parameters :
1. tournamentID – tournament ID of the tournament the round belongs to
2. roundID – round ID of the round for the heat to be stopped</td></tr>
<tr><td colspan="2">Response:
Success
{
“Stop current round heat”
}

Error
{
“An error occurred.”
}</td><td rowspan="2">Result
will just give status base on the Post. All successful POST of this API leads to this response.</td></tr>
<tr><td colspan="2">Comments:</td></tr>
</table>

Sample Payload of POST: /jsonapi/stop_heat_updated

```
$http.post("/jsonapi/stop_heat_updated", {
      tournamentID:  123456,
      roundID: 1111
   }).success(function (data, status, headers, config) {
      $scope.stopHeat_response = data;
      console.log(data);
      if (data.failed){
        alert(data.failed);
      }
      else{
        console.log("Stop current Round Heat")
        console.log(data);
        alert("Heat is stopped");
      }
   }).error(function (data, status, headers, config) {
      console.log("Error");
      alert("An error occurred.")
      console.log(data);
   });
```

Sample Output of POST: /jsonapi/stop_heat_updated

```
{"Stop current Round Heat"}
```

5. Tournament – Join

Type: GET	**Description:** Gets tournament details
URL: /jsonapi/tournament/<tournamentID> or /jsonapi/fetch_tournament/<tournamentID>	
Parameters: -	**Payload : -**

Response:


```
{
   "status": "Open for registration",
   "currentPlayerID": 57754,
   "description": "Test Group Tournament",
   "directorID": 57754,
   "isGroup": true,
   "tournamentID": 5060388987076609,
   "maxPlayersPerGroup":5,
   "maxGroups":20,
   "assignMentorInTeam": true,
   "rounds": [
      {
         "roundID": 4912161075757056,
         "currentHeatDetails": {
           "gameIDsForHeat": {
              "57754": 6096747415732224,
              "2739102": 6412161224015872
           },
           "problemsInHeat": [
              10033,
              17155
           ],
           "heatID": 4691468476219391,
           "description": "Heat 2",
           "startTime": "2013-10-04 14:02:05.835670",
           "currentTime": "2013-10-04 14:02:00.246270",
           "solvedProblemIDListsByPlayerID": {
              "57754": [
```


status
Tournament status
currentPlayerID
the current player who is logged in ID number
description
Tournament description
directorID
Tournament Director ID number
maxPlayersPerGroup
The maximum number of player to be in a group
maxGroups
The maximum number of group in the tournament
assignMentorInTeam
Mentor to be in assign within the group
rounds
Array of rounds of the tournament have
roundID
The maximum number of player to be in a group
currentHeatDetails
The details of the current ongoing heat in the round
gameIDsForHeat
game ID number for the heat for the players
problemsInHeat
problems ID number that is in the current heat
heatID
The Heat ID number
description
the heat description
startTime

```
            "/problem_is_solved_for_game/6096747415732224/10033",
            "/problem_is_solved_for_game/6096747415732224/17155"
        ],
        "2739102": [
            "/problem_is_solved_for_game/6412161224015872/10033",
            "/problem_is_solved_for_game/6412161224015872/17155"
        ] },
    "problemIDs": [
      10033,
      17155
    ],
    "heats": [
      {
        "gameIDsForHeat": {
          "57754": 5817368383062016
        },
        "problemsInHeat": [
          10033,
          17155
        ],
        "heatID": 4691468476219392,
        "description": "Heat 1",
        "startTime": "2013-09-29 08:24:46.840830",
        "currentTime": "2013-10-04 13:57:28.164880",
        "solvedProblemIDListsByPlayerID": {
          "57754": [
            "/problem_is_solved_for_game/5817368383062016/10033",
            "/problem_is_solved_for_game/5817368383062016/17155"
          ]
        },
        "stopTime": "2013-09-29 09:24:46.840830",
        "heatNumber": 1
      },
    ],
    "description": "Round 1",
    "currentHeatID": 4691468476219392,
```


the time of which the heat started

currentTime

The time in which the server return the API

solvedProblemIDListsByPlayerID

The problems which have been solved by each player in the heat

problemIDs

The problem ID number in the heat

heats

Array of previous heats details that have been conducted(object have similar properties with currentHeatDetails except with stopTime)

stopTime

The time of which the heat stop

description

Round description

currentHeatID

The ID number of the current active Heat

currentHeat

The number in which the current heat is in

problemDetails

The exact problem description

registeredPlayerIDs

Array of player that have been registered into the tournament

playerID

The registered player ID numbers

nickname

The name of the registered player

group

The group number in which the registered player is in

numRounds

The number of rounds that the tournament has

shortTitle

The title of the tournament

tournamentType

```
        "currentHeat": 2,
        "problemDetails": {
          "10033": {
            "name": "Expected Results",
            "description": "When you run your code, SingPath has certain tests that it checks to see if you did what you were supposed to do.  If incorrect you'll see a table with the results of the tests.  \r\n\r\nNotice that the starter code has the wrong value. Go ahead and run the code now so you can see the results (you can finish reading this after you run it).\r\n\r\nThe results tell you that SingPath looked at the variable named 'oops' expecting to find 713, but found 317 instead. Fix the error now and run it to advance to the next problem."
          },
          "17155": {
            "name": "Variables",
            "description": "Variables are an important part of programming; they allow the you to store a value and reuse it later.  You are allowed to use just about anything you like as your variable name.\r\n\r\nFor this problem, create a variable named 'age' with the value 7."
          }
        },
        "heatIDs": [
          4691468476219392,
          6144086545268736
        ]
      }
],
        "registeredPlayerIDs":[
                {
                        "playerID":57754,
                        "nickname":"Chris",
                        "group":1
                }],

        "numRounds": 1,
        "shortTitle": "Test Group Tournament",
        "tournamentType": "Genshyft",
        "winnerText": ""
      }
```


Default type to Genshyft to recognise as new tournaments in Singpath

winnerText

The winner text to be shown. Currently not used.

Comments:	

Sample Output of GET: /jsonapi/tournament/<tournamentID> or /jsonapi/fetch_tournament/<tournamentID>


```
{
   "status": "Open for registration",
   "currentPlayerID": 57754,
   "description": "Test Group Tournament",
   "directorID": 57754,
   "isGroup": true,
   "tournamentID": 5060388987076609,
          "maxPlayersPerGroup":5,
          "maxGroups":20,
   "assignMentorInTeam": true,
   "rounds": [
      {
         "roundID": 4912161075757056,
         "currentHeatDetails": {},
         "problemIDs": [
            10033,
            17155
         ],
         "heats": [
            {
               "gameIDsForHeat": {
                  "57754": 5817368383062016
               },
               "problemsInHeat": [
                  10033,
```

```
      17155
    ],
    "heatID": 4691468476219392,
    "description": "Heat 1",
    "startTime": "2013-09-29 08:24:46.840830",
    "currentTime": "2013-10-04 13:57:28.164880",
    "solvedProblemIDListsByPlayerID": {
      "57754": [
        "/problem_is_solved_for_game/5817368383062016/10033",
        "/problem_is_solved_for_game/5817368383062016/17155"
      ]
    },
    "stopTime": "2013-09-29 09:24:46.840830",
    "heatNumber": 1
  },
  {
    "gameIDsForHeat": {
      "57754": 6096747415732224,
      "2739102": 6412161224015872
    },
    "problemsInHeat": [
      10033,
      17155
    ],
    "heatID": 6144086545268736,
    "description": "Heat 2",
    "startTime": "2013-10-04 14:02:05.835670",
    "currentTime": "2013-10-04 13:57:28.306810",
    "solvedProblemIDListsByPlayerID": {
      "57754": [
        "/problem_is_solved_for_game/6096747415732224/10033",
        "/problem_is_solved_for_game/6096747415732224/17155"
```

```
            ],
            "2739102": [
              "/problem_is_solved_for_game/6412161224015872/10033",
              "/problem_is_solved_for_game/6412161224015872/17155"
            ]
          },
          "stopTime": "2013-10-04 15:02:05.835670",
          "heatNumber": 2
        }
      ],
      "description": "Round 1",
      "currentHeatID": 4691468476219392,
      "currentHeat": 2,
      "problemDetails": {
        "10033": {
          "name": "Expected Results",
          "description": "When you run your code, SingPath has certain tests that it checks to see if you did what you were supposed to do.  If incorrect you'll see a table with the results of the tests.  \r\n\r\nNotice that the starter code has the wrong value. Go ahead and run the code now so you can see the results (you can finish reading this after you run it).\r\n\r\nThe results tell you that SingPath looked at the variable named 'oops' expecting to find 713, but found 317 instead. Fix the error now and run it to advance to the next problem."
        },
        "17155": {
          "name": "Variables",
          "description": "Variables are an important part of programming; they allow the you to store a value and reuse it later.  You are allowed to use just about anything you like as your variable name.\r\n\r\nFor this problem, create a variable named 'age' with the value 7."
        }
      },
      "heatIDs": [
        4691468476219392,
        6144086545268736
      ]
    },
```

```
{
  "roundID": 4912161075757056,
  "currentHeatDetails": {
    "gameIDsForHeat": {
      "57754": 6096747415732224,
      "2739102": 6412161224015872
    },
    "problemsInHeat": [
      10033,
      17155
    ],
    "heatID": 4691468476219391,
    "description": "Heat 2",
    //"startTime": "2013-10-04 14:02:05.835670",
    "startTime": "2013-10-04 00:00:00",
    "currentTime": "2013-10-04 14:02:00.246270",
    "solvedProblemIDListsByPlayerID": {
      "57754": [
        "/problem_is_solved_for_game/6096747415732224/10033",
        "/problem_is_solved_for_game/6096747415732224/17155"
      ],
      "2739102": [
        "/problem_is_solved_for_game/6412161224015872/10033",
        "/problem_is_solved_for_game/6412161224015872/17155"
      ]
    },
    //"stopTime": "2013-10-04 15:02:05.835670",
    "stopTime": "2013-10-04 00:00:00",
    "heatNumber": 3
  },
  "problemIDs": [
    10033,
```

```
    17155
  ],
  "heats": [
    {
      "gameIDsForHeat": {
        "57754": 5817368383062016
      },
      "problemsInHeat": [
        10033,
        17155
      ],
      "heatID": 4691468476219391,
      "description": "Heat 1",
      "startTime": "2013-09-29 08:24:46.840830",
      "currentTime": "2013-10-04 13:57:28.164880",
      "solvedProblemIDListsByPlayerID": {
        "57754": [
          "/problem_is_solved_for_game/5817368383062016/10033",
          "/problem_is_solved_for_game/5817368383062016/17155"
        ]
      },
      "stopTime": "2013-09-29 09:24:46.840830",
      "heatNumber": 1
    },
    {
      "gameIDsForHeat": {
        "57754": 6096747415732224,
        "2739102": 6412161224015872
      },
      "problemsInHeat": [
        10033,
        17155
```

```
        ],
        "heatID": 6144086545268735,
        "description": "Heat 2",
        "startTime": "2013-10-04 14:02:05.835670",
        "currentTime": "2013-10-04 13:57:28.306810",
        "solvedProblemIDListsByPlayerID": {
          "57754": [
            "/problem_is_solved_for_game/6096747415732224/10033",
            "/problem_is_solved_for_game/6096747415732224/17155"
          ],
          "2739102": [
            "/problem_is_solved_for_game/6412161224015872/10033",
            "/problem_is_solved_for_game/6412161224015872/17155"
          ]
        },
        "stopTime": "2013-10-04 15:02:05.835670",
        "heatNumber": 2
      }
    ],
    "description": "Round 2",
    "currentHeatID": 614408654526873,
    "currentHeat": 2,
    "problemDetails": {
      "10033": {
        "name": "Expected Results",
        "description": "When you run your code, SingPath has certain tests that it checks to see if you did what you were supposed to do.  If incorrect you'll see a table with the results of the tests.  \r\n\r\nNotice that the starter code has the wrong value. Go ahead and run the code now so you can see the results (you can finish reading this after you run it).\r\n\r\nThe results tell you that SingPath looked at the variable named 'oops' expecting to find 713, but found 317 instead. Fix the error now and run it to advance to the next problem."
      },
      "17155": {
        "name": "Variables",
```

```
            "description": "Variables are an important part of programming; they allow the you to store a value and reuse it later.  You are allowed to use just about anything you like as your variable name.\r\n\r\nFor this problem, create a variable named 'age' with the value 7."
          }
        },
        "heatIDs": [
          4691468476219392,
          6144086545268736
        ]
      }],
        "registeredPlayerIDs":[
            {
                "playerID":57754,
                "nickname":"Chris",
                "group":1
            },

            {
                "playerID":2739102,
                "nickname":"James",
                "group":0
            },
            {
                "playerID":9379339,
                "nickname":"Player 3",
                "group":1
            },{
                "playerID":57753,
                "nickname":"Player 4",
                "group": 1
            },

            {
```

```
        "playerID":2739101,
        "nickname":"Player 5",
        "group":1
},
{
        "playerID":9379338,
        "nickname":"Player 6",
        "group":2
},
{
        "playerID":57752,
        "nickname":"Player 7",
        "group": 2
},

{
        "playerID":2739100,
        "nickname":"Player 8",
        "group":3
},
{
        "playerID":9379337,
        "nickname":"Player 9",
        "group":3
},
{
        "playerID":57751,
        "nickname":"Player 10",
        "group":4
},

{
```

```
                "playerID":2739099,
                "nickname":"Player 11",
                "group":4
        },
        {
                "playerID":9379336,
                "nickname":"Player 12",
                "group":5
        },
        {
                "playerID":57750,
                "nickname":"Player 13",
                "group": 5
        },

        {
                "playerID":2739098,
                "nickname":"Player 14",
                "group":5
        },
        {
                "playerId":9379335,
                "nickname":"Player 15",
                "group":5
        }],

  "numRounds": 1,
  "shortTitle": "Test Group Tournament",
  "tournamentType": "Genshyft",
  "winnerText": ""
}
```

<table>
<tr><td colspan="2">Type: POST</td><td>Description:
Join tournament group</td></tr>
<tr><td colspan="3">URL: /jsonapi/join_group/join</td></tr>
<tr><td>Payload :
{
'playerID':integer,
'tournamentID':integer,
'group':integer
}</td><td colspan="2">Description:
1. playerID is the current player ID number
2. tournamentID is the current tournament ID number
3. group is the group number the player is joining</td></tr>
<tr><td colspan="2">Response:
{ "message": String }</td><td rowspan="2">Result
will just give status base on the Post. All successful POST of this API leads to this response.
message
Error message will be shown is error occurs else success message</td></tr>
<tr><td colspan="2">Comments:
In an individual tournament group will be set as 0. To remove user from tournament it will set group to -1. Else it will be 0 for players without groups or any positive value of the group that player click to join.</td></tr>
</table>

Sample Payload of POST : /jsonapi/join_group/join

```
var data = {
       'playerID':playerId,
       'tournamentID':tournamentId,
       'group':groupNo
     };
     $scope.joining_grp = $resource('/jsonapi/join_group/join/');
     var joingrp = new $scope.joining_grp(data);
```

Sample Output of POST: /jsonapi/join_group/join

```
{"message":"success "})
```

6. Tournament Ranking and Mentor Assignment

<table>
<tr><td colspan="2">Type: GET</td><td>Description:
Use to fetch the ranking of the heat. It is also use to determine the mentor assignment</td></tr>
<tr><td colspan="3">URL: /jsonapi/get_heat_ranking?heatID=<heatID> or /jsonapi/get_heat_ranking?heatID=<heatID>&nocache=true</td></tr>
<tr><td colspan="2">Parameters: heatID, nocache</td><td>Payload : No payload. Is taken from the URL.</td></tr>
<tr><td colspan="2">Response:
{
"ranking": [
{
"status": String,
"mentee": String,
"playerid": integer,
"solved_problems": integer,
"flagUrl": String,
"finished": String
"gravatar": String
"mentor": String,
"mentorID": integer,
"professional": String,
"total_problems": integer,
"nickname": String,
"menteeID": integer,
"mentorHasArrived": boolean,
"rankingInGrp": integer,
"group": integer
}],
"heatStopTime": String,
"heatDescription": String,
"tournamentID": integer,
"tournamentDescription":String,
"roundID": integer,
"heatID": integer,
"heatStartTime": String,</td><td>Ranking
Array of player object in ranking order
status
Status of the game that the player is playing
mentee
The player mentee name who he/she will be mentoring
playerid
Player ID of the player in that object
solved_problem
Number of problems solved
flagUrl
Image file of the country flag of the person is in
gravatar
The player profile picture Gravatar URL link
mentor
The player mentor name in the game
mentorID
The player mentor’s ID
professional
the player profession status
total_problems
The total number of problems in the heat
nickname
The player name
menteeID
The player mentee’s ID
mentorHasArrived</td></tr>
</table>

"roundDescription": String, "currentTime": String, "tournamentType": String, "type": String, "isGroup": boolean, "tournamentStatus":String }	A Boolean expression to check whether have mentor arrived **rankingInGrp** In a group tournament it will reflect the player ranking in his own group. Else it will be shown as 0. **group** In a group tournament it will reflect the group number the player is in. Else it will be shown as 0. **heatStopTime** The stop time of the heat **heatDescription** The heat title description **tournamentID** The tournament ID number in which the heat is in **tournamentDescription** The tournament description **roundID** The round ID number which the heat is in **heatID** The heat ID number **heatStartTime** The start time of the heat **roundDescription** The round description **currentTime** The current time the server is retrieving the API **tournamentType** To check the tournament creation is which version. Current default type will always be “Genshyft” **type** The type of API which is returning **isGroup** A Boolean expression to identify whether is it a group or individual tournament **tournamentStatus** The current status of the tournament
Comments: API is used in both ranking and mentor assignment. (TournamentGameController.js and tournament-controller.js) */jsonapi/get_heat_ranking?heatID=<heatID> : Returns API with a 5 sec cache in the server* */jsonapi/get_heat_ranking?heatID=<heatID>&nocache=true : Returns API with the latest in the Singpath DB.*	

Sample Output of GET: /jsonapi/get_heat_ranking?heatID=4691468476219392 or /jsonapi/get_heat_ranking?heatID=4691468476219392&nocache=true


```
{
                "ranking": [
                  {
                    "status": "GAME CLOSED",
                    "mentee": "Fu Mei",
                    "playerid": 57754,
                    "solved_problems": 10,
                    "flagUrl": "/static/flags/sg_on.png",
                    "finished": "0:13:27.311930",
                    "gravatar": "http://www.gravatar.com/avatar/3f0dd6b43fe16552168c919acfbf140d/?default=&amp;s=30",
                    "mentor": "Glen",
                    "mentorID": 1111,
                    "professional": null,
                    "total_problems": 10,
                    "nickname": "Chris",
                    "menteeID": 6474597901795328,
                    "mentorHasArrived": false,
                    "rankingInGrp":1,
                    "group": 1
                  },
                  {
                    "status": "GAME CLOSED",
```

```
    "mentee": "Mrchamp",
    "playerid": 6147204892852224,
    "solved_problems": 10,
    "flagUrl": "/static/flags/sg_on.png",
    "finished": "0:15:26.809400",
    "gravatar": "http://www.gravatar.com/avatar/949ba044f4d170b60c31461607e8cb99/?default=&amp;s=30",
    "mentor": "",
    "mentorID": null,
    "professional": null,
    "total_problems": 10,
    "nickname": "Ronald",
    "menteeID": 6715360346636288,
    "mentorHasArrived": false,
    "rankingInGrp":2,
    "group": 1
  },
  {
    "status": "GAME CLOSED",
    "mentee": "Cheuk",
    "playerid": 21354567,
    "solved_problems": 10,
    "flagUrl": "/static/flags/jp_on.png",
    "finished": "0:19:10.168820",
    "gravatar": "http://www.gravatar.com/avatar/2e141f02eb0fc1b80d90c3546e4aa3e1/?default=&amp;s=30",
    "mentor": "",
    "mentorID": null,
    "professional": false,
    "total_problems": 10,
    "nickname": "Jifei",
    "menteeID": 4685146485686272,
    "mentorHasArrived": false,
    "rankingInGrp":1,
```

```
      "group": 2
    },
    {
      "status": "GAME CLOSED",
      "mentee": "Zoey",
      "playerid": 5802887565279232,
      "solved_problems": 10,
      "flagUrl": "/static/flags/sg_on.png",
      "finished": "0:21:53.351620",
      "gravatar": "http://www.gravatar.com/avatar/bc6a8b2060d57f8df5177778c7f85cb7/?default=&amp;s=30",
      "mentor": "",
      "mentorID": null,
      "professional": false,
      "total_problems": 10,
      "nickname": "Ben Chan",
      "menteeID": 5170357531049984,
      "mentorHasArrived": false,
      "rankingInGrp":2,
      "group": 2
    },
],
"heatStopTime": "2013-12-11 19:31:21.814470",
"heatDescription": "Heat 1",
"tournamentID": 5060388987076608,
"tournamentDescription": "Prize round for Python Enrichment December 2013",
"roundID": 4912161075757056,
"heatID": 4691468476219392,
"heatStartTime": "2013-12-09 07:31:21.814470",
"roundDescription": "Prize Round",
"currentTime": "2014-01-09 09:38:59.184880",
"tournamentType": "Normal",
"type": "heat ranking",
```

```
    "isGroup":true,
    "tournamentStatus":"Closed"
}
```

<table>
<tr><td colspan="2">Type: GET</td><td>Description:
Use to remove mentor and only be able to be use by the tournament director</td></tr>
<tr><td colspan="3">URL: /jsonapi/remove_mentor/:heatID/:playerID</td></tr>
<tr><td colspan="2">Parameters: heatID, playerID</td><td>Payload : No payload. Variable is taken from the URL.</td></tr>
<tr><td colspan="2">Response:
{message: String}</td><td rowspan="2">message
Error message return if error occur else success</td></tr>
<tr><td colspan="2">Comments:
API is created as a GET function instead of a POST (tournament-controller.js)</td></tr>
</table>

<table>
<tr><td colspan="2">Type: GET</td><td>Description:
Use to accept mentor on behalf of the player. Only able to be done by the tournament director</td></tr>
<tr><td colspan="3">URL: /jsonapi/accept_for_mentee/:heatID/:playerID</td></tr>
<tr><td>Parameters: heatID, playerID</td><td colspan="2">Payload : No payload. Variable is taken from the URL.</td></tr>
<tr><td colspan="2">Response:
{message: String}</td><td rowspan="2">message
Error message return if error occur else success</td></tr>
<tr><td colspan="2">Comments:
API is created as a GET function instead of a POST (tournament-controller.js)</td></tr>
</table>

<table>
<tr><td colspan="2">Type: POST</td><td>Description:
To set mentor arrival in the Singpath DB to True</td></tr>
<tr><td colspan="3">URL: /jsonapi/mentor_has_arrived</td></tr>
<tr><td>Payload :
{
'playerID':integer,
'heatID':integer
}</td><td colspan="2">Description:
1. playerID represent the current player ID
2. heatID represent the current heat which the player is playing</td></tr>
<tr><td colspan="2">Response:
{
‘message’: String
}</td><td rowspan="2">Result
Error message will be return if data POST is incorrect else a success message will be return.</td></tr>
<tr><td colspan="2">Comments:
API is used in TournamentGameController.js</td></tr>
</table>

Sample Payload of POST : /jsonapi/mentor_has_arrived

```
$scope.mentor_arrived =function(playerID, heatID){
    console.log("mentor_arrived : heatID=" + heatID);
    var data = {
     'playerID':playerID,
     'heatID':heatID
    };
    $scope.mentor_arrival = $resource('/jsonapi/mentor_has_arrived');
    var hasArrived = new $scope.mentor_arrival(data);
    hasArrived.$save(function(response){
     if(response.error) {
       console.log(response.error);
     }else{
       console.log(response);
     }
    });
    $scope.get_mentor_once(heatID, playerID);
    if($scope.mentor_hasArrived == false){
     console.log("retrieving mentor again");
     $scope.timeoutVarMentor = $timeout(function(){$scope.mentor_arrived(playerID, heatID);}, 5000);
    }else if($scope.mentor_hasArrived == true){
     $timeout.cancel($scope.timeoutVarMentor);
    }
   }
```

Sample Output of POST : /jsonapi/mentor_has_arrived / - *the expected output for the POST method*

{'playerID':577645,'heatID':49222658423}

7. Events

<table>
<tr><td colspan="2">Type: GET</td><td>Description:
Locks event ranking</td></tr>
<tr><td colspan="3">URL: /jsonapi/lock_event_ranking</td></tr>
<tr><td>Parameters:
id</td><td colspan="2">Payload :
Id - Id of event</td></tr>
<tr><td colspan="2">Response:

Not needed</td><td rowspan="2">ID
Unique id for the event</td></tr>
<tr><td colspan="2">Comments:
Can only lock once, unable to unlock</td></tr>
</table>

Sample Output of GET : /jsonapi/lock_event_ranking***)***

Not needed

<table>
<tr><td colspan="2">Type: GET</td><td>Description:
Gets event details with the specified eventID</td></tr>
<tr><td colspan="3">URL: /jsonapi/event/{eventID}</td></tr>
<tr><td colspan="2">Parameters:
Id</td><td>Payload :
Id - Event id</td></tr>
<tr><td colspan="2">Response: example of the response body – include the name and type

{
"cutoff": int,
"id": long,
"archived": boolean,
"time_to_cutoff": object,
"start": dateTime,
"editor": int,
"latitude": unknown,
"ranking": array,
"description": String,
"watching": int,
"cutoffdate": dateTime,
"registered": int,
"path": String,
"accepted": int,
"rankinglocked": boolean,
"name": String,
"created": dateTime,
"invited": int,
"venue": String,
"longitude": unknown,
"participating": int,
"following": int,
"responded": int
}</td><td>cutoff
The amount of participants the event is for
Id
eventID
archived
if true it means event has been deleted and it should not be seen anywhere in SingPath
time_to_cutoff
remaining time left till event is locked automatically
start
not in use (for countdown)
editor
event creator id
latitude
see longitude
ranking
array of participants information
description
event description ranking locked – if true it means no one else can participate in the event
watching
number of users who clicked on ‘I’d like to come watch’
cutoffdate
if set it will trigger the countdown
registered
as long as user clicks on any one of the three buttons</td></tr>
</table>

Comments: *is there anything that needs to be noted?* If eventID not specified, it will return each and every event, else it will only return that specific event's details	'participant', 'watch', keep me posted', they are considered registered **path** event path (programming language) **accepted** number of participants who have accepted the invitation and indicated that they will participate **rankinglocked** true if ranking is locked; will not allow any more users to participate **name** event name **created** when the event was created **invited** number of invites sent out **venue** venue of event **longitude** not in use **participating** number of users who clicked on "I'd like to participate **following** number of users who clicked on "keep me posted" **responded** number of invited participants who rsvp-ed

Sample Output of GET: /jsonapi/event/6095188913029120

"cutoff": 40,
"id": 6095188913029120,
"archived": false,

"time_to_cutoff": {},
"start": "2014-03-29T10:49:55.721590",
"editor": 57754,
"latitude": null,
"ranking": [],
"description": "Come join us for the National Singapore JC and High-school Coding Competition that will take place on the morning of March 29th, at 9am, at SMU. The top 40 students who have registered their school and starting year will be invited. So register now and then go solve a few Python problems. The grand prize for this event will be a MacBook Air.",
"watching": 5,
"cutoffdate": "2014-03-22T10:48:27.878890",
"registered": 85,
"path": "Python",
"accepted": 40,
"rankinglocked": true,
"name": "National Singapore JC and High-school Coding Competition",
"created": "2014-01-10T15:41:36.848030",
"invited": 40,
"venue": null,
"longitude": null,
"participating": 74,
"following": 6,
"responded": 40
}

<table>
<tr><td colspan="2">Type: GET</td><td>Description:
Respond to invitation</td></tr>
<tr><td colspan="3">URL: /jsonapi/eventrsvp/{eventID}/{decisionNum}</td></tr>
<tr><td>Parameters:
eventID
decisionNum</td><td colspan="2">Payload :
eventID – ID of event
decisionNum – 0 or 1, 0 for reject 1 for accept</td></tr>
<tr><td colspan="2">Response:

Thank you for confirming that you will be attending.

OR

Thank you for confirming that you will NOT be attending.</td><td rowspan="2">N.A.</td></tr>
<tr><td colspan="2">Comments:
No need to pass playerID, when entered into URL, will take the playerID that is currently logged in to Singpath</td></tr>
</table>

Sample Output of GET: /jsonapi/eventrsvp/6095188913029120/1

Thank you for confirming that you will be attending. **OR**

Thank you for confirming that you will NOT be attending.