All the Urls using Dynamo DB

POST /PUT Examples are at the end of this document

1. **Course Resource (courseId is a String)**

GET: ../webapi/courses/allcourses	list all courses
GET: ../webapi/courses/{courseId}	list one course by courseId
GET: ../webapi/courses/professor/{professorId}	list all courses teached by a professor id
GET: ../webapi/courses	using QueryParam studentTA - list courses belongs to a TA
GET: ../webapi/courses/{courseId}/professor	list the professor's information of a course e.g.: ../webapi/courses/CSYE6225/professor
GET: ../webapi/courses/{courseId}/TA	list the TA's information of a course e.g.: ../webapi/courses/CSYE6225/TA
DELETE: ../webapi/courses/{courseId}	delete a course
POST: ../webapi/courses	add a course
PUT: ../webapi/courses/{courseId}	update a course
PUT: ../webapi/courses/{courseId}/TA	update the TA's information of a course (request body should be a student object and you can just give it a studentId)
PUT: ../webapi/courses/{courseId}/professor	update the professor's information of a course (request body should be a professor object and you can just give it a professorId)

2. **Course Roster Resourse**

GET: ../webapi/courses/{courseId}/roster	get the roster information of a course
DELETE: ../webapi/courses/{courseId}/roster	clear the roster of a course (not delete, just clear. **Try not to use it, dangerous!!!**)

POST: ../webapi/courses/{courseId}/roster	add a student to the roster of a course (request body should be a student object)
PUT: ../webapi/courses/{courseId}/roster	delete a student from the roster of a course (request body should be a student object: you can just give it a studentId)

3. **Course Board Resource**

GET: ../webapi/courses/{courseId}/board	get the board information of a course

4. **Lecture Resource**

GET: ../webapi/courses/{courseId}/lectures	get all lectures of a course
DELETE: ../webapi/courses/{courseId}/lectures	delete a lecture of a course (request body should be a lecture: you can just give it a lectureId which is already exist in the memory)
POST: ../webapi/courses/{courseId}/lectures	add a lecture to a course (request body should be a lecture)
PUT: ../webapi/courses/{courseId}/lectures	update a lecture of a course (request body should be a lecture whose lectureId is already exist in the memory)

5. **Professor Resource**

GET: ../webapi/professors/allprofessors	list all professors
GET: ../webapi/professors	using QueryParam department. list all professor of a particular department
GET: ../webapi/professors/{professorId}	list a professor e.g. ../webapi/professors/gao.jia
DELETE: ../webapi/professors/{professorId}	delete a professor e.g. ../webapi/professors/bat.civ

POST: ../webapi/professors	add a new professor
PUT: ../webapi/professors/{professorId}	update a professor e.g. ../webapi/professors/don.wan

6. **Program Resourse**

GET: ../webapi/programs/allprograms	list all programs
GET: ../webapi/programs/{programId}	list a program by programId
GET: ../webapi/programs	using QueryParam program: list a program by name
GET: ../webapi/programs/{programId}/courses	list all courses of a program by programId e.g. ../webapi/programs/INSY/ courses
GET: ../webapi/programs/{programId}/students	list all students of a program by programId
DELETE: ../webapi/programs/{programId}	delete a program
DELETE: ../webapi/programs/{programId}/courses	delete a course of a program (request body should be a course object: you can just give it a courseId (it is a String such as "CYSE6225"))
POST: ../webapi/programs	add a new program
POST: ../webapi/programs/{programId}/courses	add a course to a program (request body should be a course object)
PUT: ../webapi/programs/{programId}	update a program

7. **Student Resource**

GET: ../webapi/students	using QueryParam programName: list all students of a particular program
GET: ../webapi/students/{studentId}	list a student e.g. ../webapi/students/lin.jin
DELETE: ../webapi/students/{studentId}	delete a student
POST: ../webapi/students	add a new student
PUT: ../webapi/students/{studentId}	update a student

8. **Announcement Resource**

POST: ../webapi/announcements — add a new announcement

If the length of announcementText is **more than 160**

You will get an **500 error**.

If the length of annoucementText is less than 160, you can add it to database successfully

GET: ../webapi/announcements/allannouncements — get all announcements

GET: ../webapi/announcements/{boardId}_{annId} — get an announcement

e.g. ../webapi/announcements/1_ann01

GET: ../webapi/announcements/{boardId} — get announcements by boardId

e.g. ../webapi/announcements/1

DELETE: ../webapi/announcements/{boardId}_{annId} — delete an announcement

PUT: ../webapi/announcements/{boardId}_{annId} — update an announcement

EXAMPLE: They are not all of the examples.

1. POST: ../webapi/courses — add a course

```
{
   "courseId": "CYSE1234",
   "department": "Computer Science Align",
   "lectures": [
      7
   ],
   "professorId": "fan.fel",
   "taId": "wu.jia"
}
```

2. PUT: ../webapi/courses/{courseId}/TA — update the TA's information of a course

```
{
      "studentId": "tom.han"
}
```

3. PUT: ../webapi/courses/{courseId}/professor — update the professor's information of a course

```
{
   "professorId": "fan.fel"
```

}

4. POST: ../webapi/courses/{courseId}/roster add a student to the roster of a course

```
{
        "studentId": "tom.han"
}
```

5. POST: ../webapi/courses/{courseId}/lectures add a lecture to a course

```
{
  "lectureId": 8,
  "courseId": "CYSE6225",
  "notes": [],
  "materials": []
}
```

6. POST: ../webapi/professors add a new professor

```
{
   "professorId":"wan.ton",
   "firstName":"Tong",
   "lastName":"Wang",
   "department":"Computer Science Align",
   "joiningDate":"09/01/2015"
}
```

7. PUT: ../webapi/professors/don.wan update a professor

```
{
"professorId":"don.wan",
"firstName":"Dong",
"lastName":"Wang",
"department":"Computer Science Align",
"joiningDate":"09/01/2017"
}
```

8. POST: ../webapi/programs add a new program

```
{
  "programId":"AAAA",
  "programName":"A program A",
  "students":[],
  "courses":[]
```

```
}
```

9. POST: ../webapi/programs/{programId}/courses add a course to a program

```
{
  "courseId":"AAAA"
}
```

10. POST: ../webapi/students add a new student

```
{
  "firstName":"Xiaoyuan",
  "lastName":"Fan",
  "studentId":"fan.xia",
  "imageUrl":"Do not have a url",
  "department":"Information Systems",
  "joiningDate":"09/10/2017",
  "courses": [
    "INFO7390"
  ]
}
```

11. PUT: ../webapi/students/har.ron update a student

```
{
  "courses": [
    "INFO7300",
    "INFO6205",
    "INFO5000",
    "INFO7390"
  ],
```

```
    "department": "Computer Science Align",
    "firstName": "Rong",
    "id": "a1766a70-854c-40cc-89be-bd2f4b0ecd37",
    "imageUrl": "Do not have a url",
    "joiningDate": "09/10/2016",
    "lastName": "Harsha",
    "studentId": "har.ron"
  }
```


12. POST: ../webapi/announcements add a new announcement


```
{
  "announcementId":"ann06",
  "announcementText":"This is the sixth annoucement!",
  "boardId":5
}
```


13. PUT: ../webapi/announcements/1_ann04 update a new announcement


```
{
  "announcementId":"ann04",
  "announcementText":"abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyUPDATE",
  "boardId":1
}
```