Please read the following instructions carefully.

Guidelines:

Once it is done, upload your work as *an attachment.*

Your job is to build simple crud operations with api's, which does the following:

- Add course offering
    - A course offering has course title, instructor and date.
    - It should also contain a minimum & maximum number of employees for the course offering.
- Register for the course
    - Employees can register for the courses.
    - If no. of employees registered for the course has reached the maximum, the result will be COURSE_FULL_ERROR.
    - Otherwise, result of registration will be ACCEPTED.
- Cancel registration
    - Employees can cancel their registration until the course allotment is completed.
- Course allotment
    - This feature allots employees to course offering, before the course offering date.
    - It should print a list of all the employees with their details along with their final course allotment status (Registration Number, Employee Name, Email, Course Offering ID, Course Name, Instructor, Date, Final Status). The list should be sorted based on the Registration number.
    - If sufficient registrations are not received then the course offering itself gets cancelled.
    - The employees who have registered will get confirmed unless the minimum number of registrations is not received.
    - Even if the course offering gets canceled due to the minimum number of employees not registered, the list should be printed.

Note: While working on this Backend task, you are required to use **AWS DynamoDB**.

- Please complete the pre-requisite videos of DynamoDB (Available in Developer Sheets of Attached sheet) if you have not done so already.
- You must use only one table for this task.
- The table should have both a Partition Key (PK) and a Sort Key (SK), and both are mandatory.

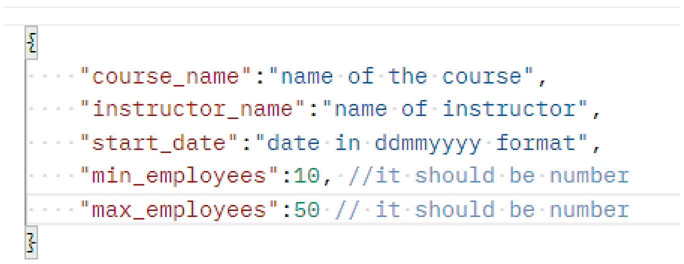
API'S

1. Add course offering

Route : /add/courseOffering

Method: POST

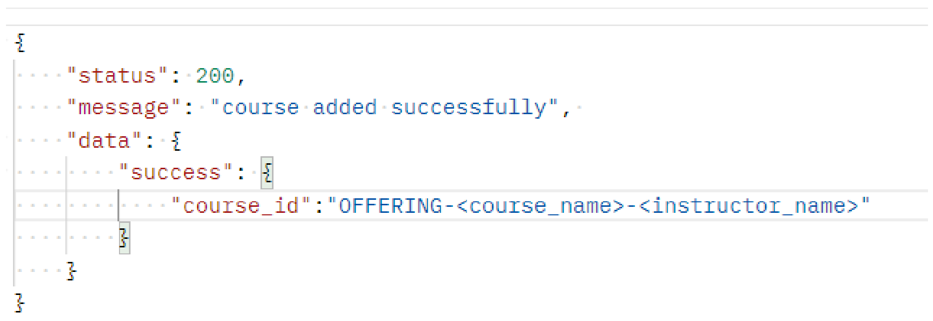
Body:



```
{
    "course_name":"name of the course",
    "instructor_name":"name of instructor",
    "start_date":"date in ddmmyyyy format",
    "min_employees":10, //it should be number
    "max_employees":50 // it should be number
}
```

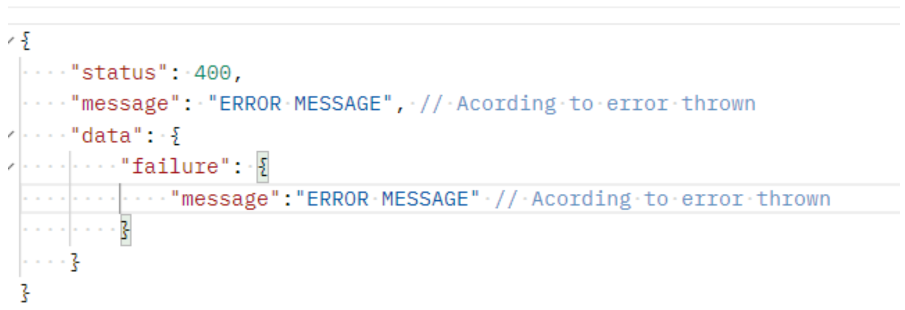
The format of course-offering-id is OFFERING-<COURSE-NAME>-<INSTRUCTOR>

Successful response:



```
{
    "status": 200,
    "message": "course added successfully",
    "data": {
        "success": {
            "course_id":"OFFERING-<course_name>-<instructor_name>"
        }
    }
}
```

Error response:

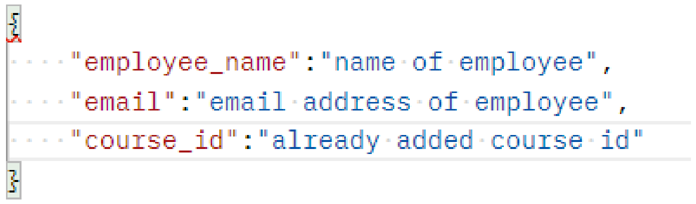


```
{
    "status": 400,
    "message": "ERROR MESSAGE", // Acording to error thrown
    "data": {
        "failure": {
            "message":"ERROR MESSAGE" // Acording to error thrown
        }
    }
}
```

2. Register for the course

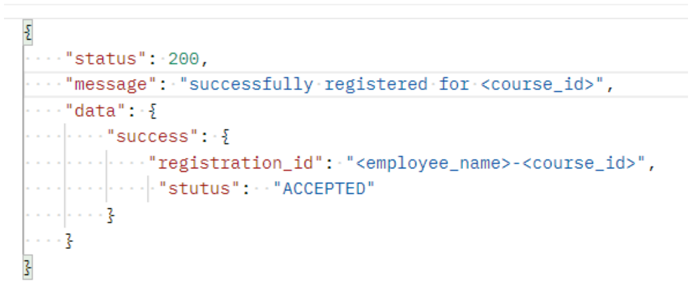
a. The combination of email-id, name and course_id in the input should be unique
b. The format of course-registration-id is <EMPLOYEE-NAME>-<COURSE-NAME>
c. If number of employees has not exceeded the maximum number of employees allowed for the course offering, status will be ACCEPTED
d. If number of employees has exceeded the maximum number of employees allowed for the course offering, status will be COURSE_FULL_ERROR
e. If the minimum number of employees for the course offering is not reached before the course date, the status of the course offering would be COURSE_CANCELED
f. Course-id will only be returned if the status is ACCEPTED

Route : /add/register/:course_id //here course_id will be dynamic



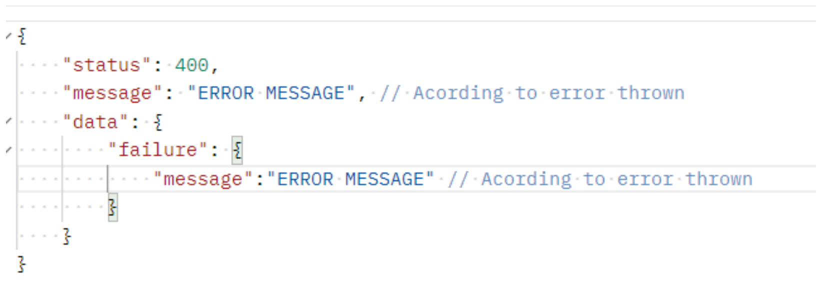
```
{
    "employee_name":"name of employee",
    "email":"email address of employee",
    "course_id":"already added course id"
}
```

Method: POST **Body**:



```
{
    "status": 200,
    "message": "successfully registered for <course_id>",
    "data": {
        "success": {
            "registration_id": "<employee_name>-<course_id>",
             "stutus":  "ACCEPTED"
        }
    }
}
```

Successful response:



```
{
    "status": 400,
    "message": "ERROR MESSAGE", // Acording to error thrown
    "data": {
        "failure": {
            "message":"ERROR MESSAGE" // Acording to error thrown
        }
    }
}
```

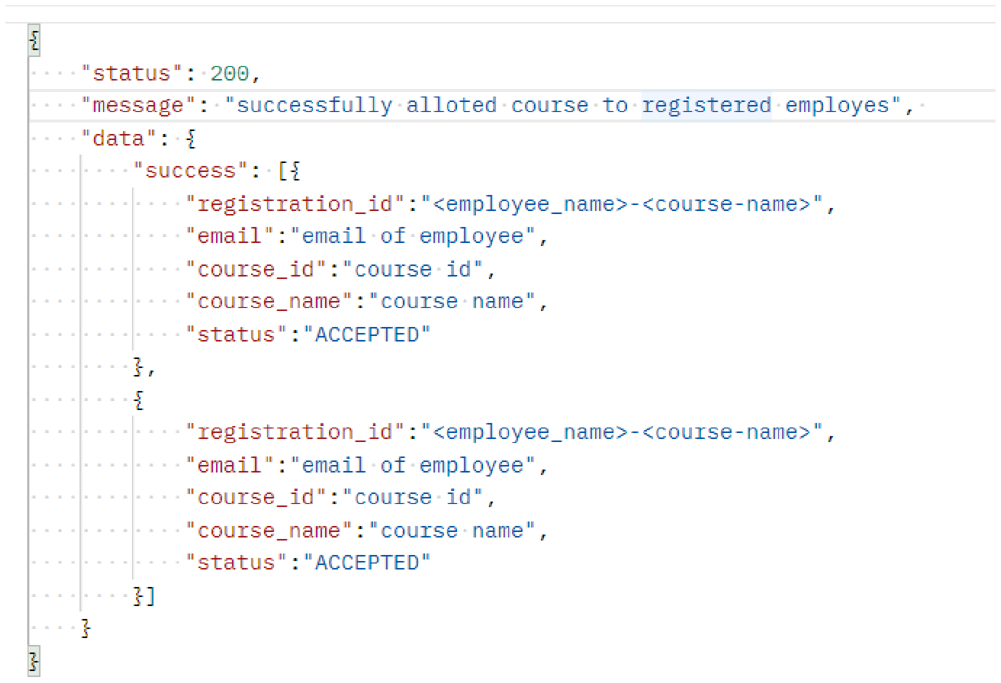
Error response:

3. Course allotment

1. The output should be sorted by course-registration-id in ascending order

Route : /allot/:course_id

Method: POST

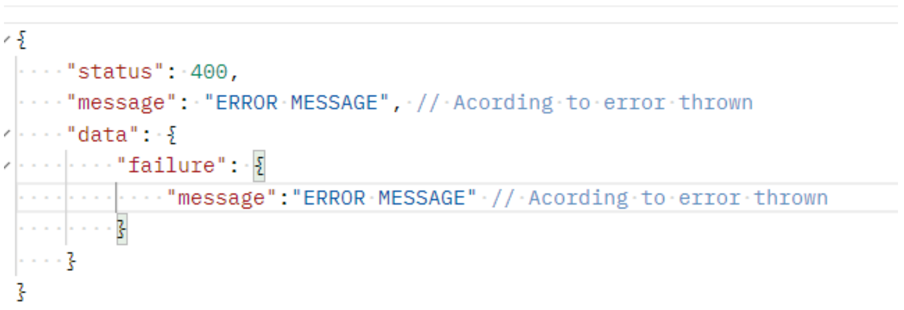


```
{
    "status": 200,
    "message": "successfully alloted course to registered employes",
    "data": {
        "success": [{
            "registration_id":"<employee_name>-<course-name>",
            "email":"email of employee",
            "course_id":"course id",
            "course_name":"course name",
            "status":"ACCEPTED"
        },
        {
            "registration_id":"<employee_name>-<course-name>",
            "email":"email of employee",
            "course_id":"course id",
            "course_name":"course name",
            "status":"ACCEPTED"
        }]
    }
}
```

Body:

NO body

Successful response:



```
{
    "status": 400,
    "message": "ERROR MESSAGE", // Acording to error thrown
    "data": {
        "failure": {
            "message":"ERROR MESSAGE" // Acording to error thrown
        }
    }
}
```

Error response:

4. Cancel the course registration:

The employee can cancel the course registration given a course registration id until the course allotment

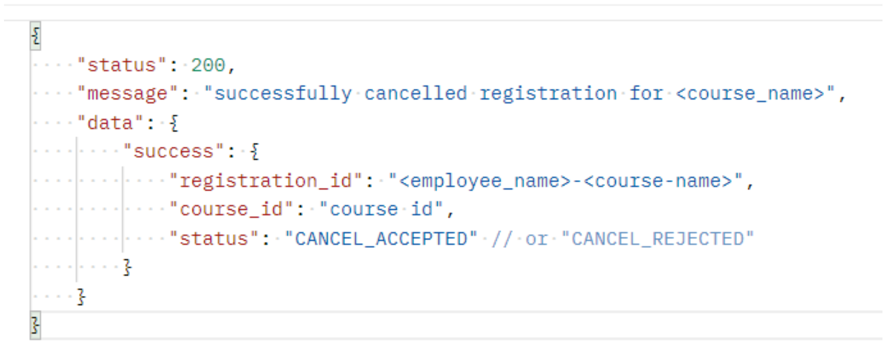
- There are 2 statuses: CANCEL_ACCEPTED, CANCEL_REJECTED

- CANCEL_ACCEPTED when the cancellation is successful.
- CANCEL_REJECTED when the course is already allotted.

Route : /cancel/:registration_id

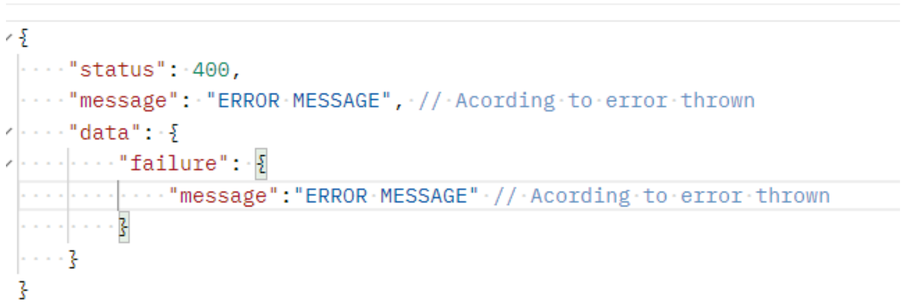
Method: POST

Body: NO body



```
{
    "status": 200,
    "message": "successfully cancelled registration for <course_name>",
    "data": {
        "success": {
            "registration_id": "<employee_name>-<course-name>",
            "course_id": "course id",
            "status": "CANCEL_ACCEPTED" // or "CANCEL_REJECTED"
        }
    }
}
```

Successful response:



```
{
    "status": 400,
    "message": "ERROR MESSAGE", // Acording to error thrown
    "data": {
        "failure": {
            "message":"ERROR MESSAGE" // Acording to error thrown
        }
    }
}
```

Error response:

- There are 2 statuses : CANCEL_ACCEPTED, CANCEL_REJECTED
- CANCEL_ACCEPTED when the cancellation is successful.
- CANCEL_REJECTED when the course is already allotted.

Input / Output

1. **ADD-COURSE-OFFERING**

Input	**output**
{ course_name:”JAVA”, instructor_name:”JAMES” start_date: “15062022” min_employees: 1 max_employees:2 }	{ "status": 200, "message": "course added successfully", "data": { "success": { "course_id":"OFFERING-JAVA-JAMES " } } }
{ course_name:”PYTHON”, }	{ "status": 400, "message": "INPUT_DATA_ERROR", "data": { "success": { "failure":"instructor_name,dtart_date, min_employees, max_employees cannot be empty" } } }

	//(because of missing instructor and course-offering-date,min max epmployees)

{
course_name**:**”KUBERNETES ”,
instructor_name:”WOODY”
start_date: “16062022”
min_employees: 2
max_employees:6
}
	{
	"status": 200,
	"message": "course added successfully",
	"data": {
	"success": {
	"course_id":"OFFERING-KUBERNETES-WOODY"
	}
	}
	}

2. REGISTER FOR COURSE

input	**output**

{
employee_name:”ANDY”,
email:”ANDY@GMAIL.COM ”
course_id: “OFFERING-JAVA-JAMES”
}
	{
	"status": 200,
	"message": "successfully registered for OFFERING-JAVA-JAMES",
	"data": {
	"success": {
	"registration_id":"ANDY-OFFERING-JAVA-JAMES",
	“stutus”: “ACCEPTED”
	}
	}
	}

{
employee_name:”JHON”,
email:”JHON@GMAIL.COM ”
course_id: “OFFERING-JAVA-JAMES”
}
	{
	  "status": 200,
	  "message": "successfully registered for OFFERING-JAVA-JAMES",
	  "data": {
	    "success": {
	      "registration_id":"JHON-OFFERING-JAVA-JAMES",
	      “stutus”: “ACCEPTED”
	
	          }
	       }
	}

{
employee_name:”JACK”,
email:”JACK@GMAIL.COM ”
course_id: “OFFERING-JAVA-JAMES”
}
	{
	  "status": 400,
	  "message": "COURSE_FULL_ERROR",
	  "data": {
	    "failure": {
	      "Message":"cannot register for course, course if full"
	    }
	  }
	}

{employee_name:”JACK”}	{"status": 400,
	 "message": "INPUT_DATA_ERROR",
	  "data": {"failure": {"Message": "email and course-offering-id missing"
	    } } }

3. ALLOT Course

Input	output

/allot/OFFERING-JAVA-JAMES	{
	"status": 200,
	"message": "successfully allotted course to registered employees",
	"data": {
	"success": [
	{
	registration_id:"ANDY-OFFERING-JAVA-JAMES",
	"email":"ANDY@GMAIL.COM",
	"course_name: "JAVA",
	"course_id":"OFFERING-JAVA-JAMES ",
	"status":"ACCEPTED"
	},
	{
	registration_id:"JHON-OFFERING-JAVA-JAMES",
	"email":"JHON@GMAIL.COM",
	"course_name: "JAVA",
	"course_id":"OFFERING-JAVA-JAMES ",
	"status":"ACCEPTED"
	}
	]
	}
	}

4. CANCEL REGISTRATION

Input	output

/CANCEL/ANDY-OFFERING-JAVA-JAMES	{
	"status": 200,
	"message": "Cancel registration unsuccessfull",
	"data": {
	"success": {
	"registration_id":"ANDY-OFFERING-JAVA-JAMES ",
	"course_id":"OFFERING-JAVA-JAMES ",
	status:"CANCEL_REJECTED"
	}
	}
	}