REST API Documentation for Backend

BASE URL:

Replace all instances of example.com and localhost with BASE URL
The Backend API is split into 3 parts (3 files, 3 ports).
Each part deals with particular databases.

Files

StudentListDatabaseAPI.go runs on port - 3000
StudentProfileDatabaseAPI.go runs on port - 3001
ClassAttendanceListAPI.go runs in port – 3002

File – Database Relation

ClassAttendanceListAPI.go deals with the following databases:
- classlist
- All auto generated databases

StudentListDatabaseAPI.go deals with the following databases:
- studentlist

StudentProfileDatabaseAPI.go deals with the following databases:
- studentprofile
- bluetoothid
- All auto generated databases

Port 3000

API 1 – CreateClass

Description:
This API enables you to create a new Class in classlist table. This should only be done by the school administration. Inserting into this Database automatically creates a separate Database for every entry. These are the auto generated databases.

It will also populate the bluetoothid database appropriately.

Url - http://example.com:3002/createclass/:classid/:classname:/bluetoothid

Curl - curl -X POST "http://localhost:3002/createclass/283/Virtualization/1234"

Method - GET

Variables -
:classid – int
:classname – string
:bluetoothid – string

Response:
Body – Empty

Status : 200 OK

Error - None

API 2 – DeleteClass

Description:
This API enables you to delete a Class in classlist table. This should only be done by the school administration. The auto generated table will be automatically deleted. Corresponding entries in bluetoothid database will also get deleted.

Url - http://example.com:3002/deleteclass/:classid

Curl - curl -X DELETE "http://localhost:3002/deleteclass/283"

Method - DELETE

Variables -
:classid - int

Response:
Body – Empty

Status : 200 OK

Error - None

API 3 – ClassAttendance

Description:
This API gives you the list of students that are present in a class.

Url - http://example.com:3002/classattendance/:classid

Curl - curl -X GET "http://localhost:3002/classattendance/273"

Method - GET

Variables -
:classid - int

Response:
Body -
```
[{
	"id": "44f5a9ca7b28013494043302c4002076",
	"key": 5001,
	"value": "kou"
},
{
	"id": "44f5a9ca7b28013494043302c4045376",
	"key": 5002,
        "value": "rav"
}]
```

Body Variables -
id – string – document id
key – string – student id
value – string – student name

Status : 200 OK

Error – None

API 4 – ClearClassAttendance

Description:
This API enables you to clear the attendance list for a class once a class is over.

Url - http://example.com:3002/clearclassattendance/:classid

Curl - `curl -X DELETE "http://localhost:3002/clearclassattendance/283"`

Method - DELETE

Variables -
:classid - int

Response:
Body – Empty

Status : 200 OK

Error – None

PORT 3000

API 1 – StudentName

Description:
This API gives you the name of a student gives his/her student id.

Url - http://example.com:3000/studentname/:studentid

Curl - curl -X GET "http://localhost:3000/studentname/5001"

Method - GET

Variables -
:studentid - int

Response:
Body -
```
{
        "id": "44f5a9ca7b28013494043302c4002076",
        "key": 5001,
        "value": "kou"
}
```

Body Variables -
id – string – document id
key – string – student id
value – string – student name

Status : 200 OK

Error – None

API 2 – CheckStudentValid

Description:
This API tells you if a student is a valid student or not. Ie, if the college recognizes the student as existent (if studentid exists in studentlist database).

Url - http://example.com:3000/checkstudentvalid/:studentid

Curl - curl -X GET "http://localhost:3000/checkstudentvalid/5001"

Method - GET

Variables -
:studentid - int

Response:
Body -
```
{
	"status":"yes"
}
```

Body Variables -
status – string – Student is Valid

Status : 200 OK

Error -
```
{
	"status":"no"
}
```

Error Variables -
status – string – If “no” student is invalid

API 3 – StudentEnrolled

Description:
Given a studentid, it returns all the classid of all the subjects the student has registered to

Url - http://example.com:3000/studentenrolled/:studentid

Curl - curl -X GET "http://localhost:3000/studentenrolled/5001"

Method - GET

Variables -
:studentid - int

Response:
Body -
```
{
	"id": "44f5a9ca7b28013494043302c4002076",
	"key": 5001,
	"value": [273, 283]
}
```

Body Variables -
id – string – document id
key – int – student id
value – []int – List of classes student has registered for

Status : 200 OK

Error -
No Error

API 4 – AllStudent

Description:
Gives a list of all students that are enrolled in college.

Url - http://example.com:3000/allstudent

Curl - curl -X GET "http://localhost:3000/allstudent"

Method - GET

Variables -
None

Response:
Body -
```
{
	"total_rows": 3,
	"offset": 0,
	"rows": [{
		"id": "44f5a9ca7b28013494043302c4002076",
		"key": 5001,
		"value": "kou"
	}, {
		"id": "44f5a9ca7b28013494043302c4011959",
		"key": 5002,
		"value": "Rav"
	}, {
		"id": "44f5a9ca7b28013494043302c401290a",
		"key": 5003,
		"value": "Ish"
	}]
}
```

Body Variables -
total_rows – int – Total number of students
offset – int – Pagination Offset
id – string – Document Id
key – int – Student Id
value – string – Student Name

Status : 200 OK

Error -
No Error

API 5 – AddStudent

Description:
This API allows the college administration to add a new student to be a part of college.

Url - http://example.com:3000/addstudent

Curl - `curl -X POST "http://localhost:3000/addstudent" -d '{"studentid": 5004,"regclasses": [273, 283],"studentname": "Situ"}'`

Request Body -
```
{
	"studentid": "5004",
	"regclasses": [273, 283],
	"studentname": "Situ"
}
```

Request Body Variables -
studentid – int – Unique Id of student
regclasses – []int – List of classes tudent has registered
studentname – string – Student name

Method - POST

Variables -
None

Response:
Body -
None

Status : 201 CREATED

Error -
```
{
	"error":"Json not correct"
```

}

Error Variables -
error – string – If “Json not correct”, then Json not correct

---

PORT 3001

API 1 – RegisterStudent

Description:
This API allows a valid student(id must be in studentlist DB) to register/ sign up using an android app/client. The student must provide their student id and password while signing up.

In the response, you will get deviceid and the list of BLE uuids (corresponding to classes the student has registered) that the app/client has to keep track of.

The deviceid has to be sent to the server everytime a student wants to mark himself/herself present. The BLE uuids help the mobile client to filter out only the BLE chips corresponding to the classes the student has registered for.

The password has to be resent again with a delete request to delete the student record.

Url - http://example.com:3001/registerstudent/:studentid/:password

Curl - `curl -X POST "http://localhost:3001/registerstudent/5004/swag"`

Method - POST

Variables -
:studentid – int
:password – string

Response:
Body -
```
{
        "deviceid": "907d15d2486721382398768db0000c01",
        "bluetoothids": [“1234”, “123456”]
}
```

Response Variables -

deviceid – string – The unique device id that is supposed to be stored by client
bluetoothids - []string – List of BLE chip's uuid to keep track of

Status : 201 CREATED

Error -
```
{
	"error":"Already Registered"
}
or
{
	“error”:”Not Valid Student”
}
```

Error Variables -
error – string – If “Already Registered”, then Already Registered
                    – If “Not Valid Student”, then Not Valid Student

---

API 2 – DeleteStudent

Description:
This API allows to delete the record of an already registered student in the Database. The student also has to provide the password used during sign up.

Url - http://example.com:3001/deletestudent/:studentid/:password

Curl - `curl -X DELETE"http://localhost:3001/deletestudent/5004/swag"`

Method - DELETE

Variables -
:studentid – int
:password – string

Response:
Body -
None

Status : 200 OK

Error -
```
{
	"error":"Student does not exist"
}
```

Error Variables -
error – string – If “Student does not exist”, then   Student does not exist.

API 3 – MarkPresent

Description:
This API allows a student to mark himself/herself present for a particular class. The unique deviceid generated during the register process has to be sent along with this request to prevent proxying and provide authentication.

The student has to be a valid student and has to be registered.

Url - http://example.com:3001/markpresent/:studentid/:deviceid/:classid

Curl - `curl -X POST "http://localhost:3001/markpresent/5004/907d15d2486721382398768db00022d0/273"`

Method - POST

Variables -
:studentid – int
:deviceid – string
:classid – int

Response:
Body -
None
```
{
	"success": "Student is marked present"
}
```

Status : 200 OK

Error -

```
{
	"error":"Student already present"
}
or
{
	"error":"Student is cheating. StudentId and DeviceId dont match"
}
or
{
	"error":"Student not registered"
}
```

Error Variables -
error – string