CouchDB Structure & API Documentation

Base URL: https://uname:password@ip/

List of Databases

Databases (required):

* studentlist
* studentprofile
* classlist
* bluetoothid

Databases (Auto Generated):

* class<insert classid>

Eg. If we create a new class with classid 300, then a new database 'class300' will get automatically created

---

Database Structures and View Endpoints

classlist -

This database has details of all the classes happening in the college for the semester. It holds the class number and the name of the class This database has to be filled up by the college administration. Inserting into this Database automatically creates a separate Database for every entry (If you insert via the provided backend API). These are the Auto Generated databases that I have mentioned above.

Note: This database has to be filled up before any other database

All documents have 4 fields:

* _id	`string  -  Unique Id of the document`
* _rev	`string  -  Revision Id of the document`
* classnumber	`int  -  Course number of the course. Eg 273`
* classname	`string  -  Name of the Course`

Example Document:

```
{
  "_id": "c8bce53ea287a677199c7d8c33003ef8",
  "_rev": "1-bb82a350aa615ee84a12c30ff36a2dda",
  "classnumber": 273,
  "classname": "DistributedSystems"
}
```

Design Document:

This design document emits a key value pair of classnumber and classname.
Endpoint:
https://admin:9631aa6374e6@couchdb-80f683.smileupps.com/classlist/_design/getclassdata/_view/classexists

```
{
  "_id": "_design/getclassdata",
  "_rev": "1-7811abb20f9635ddbec007265c3da1d8",
  "views": {
     "classexists": {
        "map": "function(doc){ emit(doc.classnumber, doc.classname)}"
     }
  }
}
```

---

bluetoothid -

This database holds the details of which bluetooth sensor is for which subject. Each classid is mapped to the Unique uuid of the bluetooth sensor. The BLE sensor has to be already set up to transmit the classid using iBeacon facility. The mobile app will use this uuid information to filter out which BLE sensors to connect to. This database has to be filled up by the college administration.

Note: This database has to be filled up immediately after classlist database

All documents have 4 fields:

* _id	`string  -  Unique Id of the document`
* _rev	`string  -  Revision Id of the document`
* classid	`int  -  Course number of the course. Eg 273`
* bluetoothid	`int  -  UUID/ MAC address of the corresponding BLE`

Example Document:

```
{
  "_id": "c8bce53ea287a677199c7d8c33002574",
  "_rev": "1-f891b343d1acfccfe14aae6cccf649e1",
  "classid": 283,
  "bluetoothid": 123456
}
```

Design Document:

This design document emits a key value pair of classid and bluetoothid.
Endpoint:
https://admin:9631aa6374e6@couchdb-80f683.smileupps.com/bluetoothid/_design/getbluetoothid/_view/bluetoothid

```
{
  "_id": "_design/getbluetoothid",
  "_rev": "1-01c11f49f753ea2a8dde7d3eea0f64a9",
  "views": {
     "bluetoothid": {
        "map": "function(doc){ emit(doc.classid, doc.bluetoothid)}"
     }
  }
}
```

---

studentlist -

This database has the master list of students that are recognized by the college.
The college administration is responsible for populating it. The regclasses fields holds all the classid s to which the student has registered for the semester. Also should be provided by administration.

Note: Entries of regclasses must already exist in classlist database

All documents have 5 fields:

* _id `string - Unique Id of the document`
* _rev `string - Revision Id of the document`
* studentid `int - Unique Id of the student `
* studentname `string - Name of the student`
* regclasses `[]int - Array of classid to which the student is registered`

Example Document:

```
{
  "_id": "44f5a9ca7b28013494043302c4011959",
  "_rev": "2-4f523cae598a0880ce01ac6a76dd0258",
  "studentid": 5002,
  "regclasses": [
     273,
     283
  ],
  "studentname": "Rav"
}
```

Design Document:

This design document has 2 views.
Studentname: emits a key value pair of studentid and studentname.
studentenrolled: emits a key value pair of studentid and regclasses.

Endpoints:
https://admin:9631aa6374e6@couchdb-80f683.smileupps.com/studentlist/_design/getlistdata/_view/studentname
https://admin:9631aa6374e6@couchdb-80f683.smileupps.com/studentlist/_design/getlistdata/_view/studentenrolled

```
{
  "_id": "_design/getlistdata",
  "_rev": "3-747d21d69ca5ddeb5edf8d158542018f",
  "views": {
     "studentname": {
        "map": "function(doc){ emit(doc.studentid, doc.studentname)}"
     },
     "studentenrolled": {
        "map": "function(doc){ emit(doc.studentid, doc.regclasses)}"
     }
  }
}
```

---

studentprofile -

This database holds the list of all the students who have signed up with an Android app. During signup, the student provides a password.

Note: All the previous databases have to be set up in order to use our APIs on this database

All documents have 4 fields:
* _id   `string  -  Unique Id of the document`
* _rev   `string  -  Revision Id of the document`
* studentid   `int  -  Unique Id of the student `
* password   `string  -  Password provided by student during sign up`

Example Document:

```
{
  "_id": "c8bce53ea287a677199c7d8c330039e1",
  "_rev": "1-f59d62e90d8a222896db6578a11dff71",
  "studentid": 5001,
  "password": "yolo"
}
```

Design Document:

This design document has 2 views.
studentregistered: emits only a key studentid
studentpassword: emits a key value pair of studentid and password.

Endpoints:
https://admin:9631aa6374e6@couchdb-80f683.smileupps.com/studentprofile/_design/studentdetails/_view/studentregistered
https://admin:9631aa6374e6@couchdb-80f683.smileupps.com/studentprofile/_design/studentdetails/_view/studentpassword

```
{
  "_id": "_design/studentdetails",
  "_rev": "5-429a99dd45a2b89fc447040c8932315c",
  "views": {
     "studentregistered": {
        "map": "function(doc){ emit(doc.studentid)}"
     },
     "studentpassword": {
        "map": "function(doc){ emit(doc.studentid, doc.password)}"
     }
  }
}
```

Auto Generated Databases -

A separate database will be auto generated for every classid in the classlist database on insert into classlist database. Basically, this database will hold the list of all students that are present in the class. The design document is automatically generated as well

So every class has a separate list of the students who are present in it.

All documents have 3 fields:

```
* _id                          `string  -  Unique Id of the document`
* _rev                         `string  -  Revision Id of the document`
* studentid                     `int  -  Unique Id of the student `
```

Example Document:

```
{
  "_id": "cb8b8f928333133ae1a0e4e0e9000d85",
  "_rev": "1-2f5a207fa9a0f1a2191e801d1a2c8ae9",
  "studentid": 5001
}
```

Design Document:

This design document emits a key value pair of studentid and studentid

Endpoint:

https://admin:9631aa6374e6@couchdb-80f683.smileupps.com/studentprofile/_design/classattendance/_view/isstudentpresent

```
{
  "_id": "_design/classattendance",
  "_rev": "1-662b76fcca20327d679e8fcaf3bc395f",
  "views": {
      "isstudentpresent": {
          "map": "function(doc){ emit(doc.studentid, doc.studentid)}"
      }
  }
}
```