**ZapApi**

This REST API is made to simulate a part of an online ordering system and supports GET, POST and DELETE methods for the following three objects of the system.

* Restraunt
* Menu
* Menu Items

**Getting Started**

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes. See deployment for notes on how to deploy the project on a live system.

**Prerequisites**

In Order to get this API simulated on the local machine following are the rquirements

* Python 3
* Flask - Restful
* MySQL **The server can be run by installing the python dependencies and calling the [ ZapApi.py ] file from the python directory.**

**PYTHON CONFIG FOR MYSQL DATABASE CONNECTION REQUIREMENTS**

**app.config['MYSQL\_DATABASE\_USER'] =** 'user\_name' **app.config['MYSQL\_DATABASE\_PASSWORD']** = 'password'**app.config['MYSQL\_DATABASE\_DB']** = 'databsename' **app.config['MYSQL\_DATABASE\_HOST']** = 'localhost'

**DATABSE REQUIREMENTS**

The database requires MySQL and following are the entities that I worked with.

* **t1** : Table that stores the following attributes for the **Restraunt**
  + **name** : Name of the Restraunt
  + **city** : City in which the restraunt is located
  + **rid** : A Unique ID for the Restraunt - Which I created as first alphabet of the city followed by initials of the restraunt name and followed by a random number.
  + Example : name: Wendys ; city : Dallas ; rid: DW01
* **t3** : Table that stores the following attributes for the **MenuItems**
  + **mid** : Stores Uniques ID for the Menu Item Name - which I created as type of the menu followed by a random number
  + Example: mtype: Breakfast , mname:Pan Cakes, mid: B01
  + **mtype** : Stores the type of the menu.
  + Example : Breakfast , Lunch , Dinner
  + **mname** : Stores name of the Food Item
* **t2** : Table that store the following attributes to be referenced in the t1 and t3 table as foreign keys
  + **rid** : referenced as a foreign key in the t1 table on rid for restraunts
  + **mid** : referenced a the foreign key in the t3 table on mid for the Food Items.

The following constraints need to be defined while creating the table for the above databse. Table **t1** :

* **rid** should be **VARCHAR(45)** | **Primary key , Not NULL , Unique**
* **name** should be **VARCHAR(45)** |**not NULL**

Table **t3** :

* **mid** should be **VARCHAR(45)** | **Primary key , Not NULL , Unique**
* **mtype** should be **VARCHAR(45)**
* **mname** should be **VARCHAR(45)**

Table **t2** :

* **rid** should be **VARCHAR(45)**
* **mid** should be **VARCHAR(45)**
* Foreign Key reference on **mid** : Foreign Key Name : **FK\_mid** with **CASCADE on update** and **CASCADE on delete**|Reference table : **t3**
* Foreign Key reference on **rid** : Foreign Key Name : **FK\_rid** with **CASCADE on update** and **CASCADE on delete**| Reference table : **t1**

The following SQL procedures were called in the api for POST , GET and DELETE functions.

* **spcreate** : This procedure creates/adds a restraunt in the t1 table.
* CREATE DEFINER=`root`@`localhost` PROCEDURE `spcreate`(IN p\_name varchar(45),IN p\_city varchar(45), IN r\_id varchar(45))
* BEGIN
* if ( select exists (select 1 from t1 where name = p\_name) ) THEN
* select 'Name Exists !!';
* ELSE
* insert into t1
* (
* name,
* city,
* rid
* )
* values
* (
* p\_name,
* p\_city,
* r\_id
* );
* END IF;
* END
* **menuitem** : This procedure adds the menu item to the t3 table.
* CREATE DEFINER=`root`@`localhost` PROCEDURE `menuitem`(IN m\_type varchar(45), IN m\_id varchar(45), IN m\_name varchar(45))
* BEGIN
* if ( select exists (select 1 from t3 where mid = m\_id) ) THEN
* select 'Menu Exists !!';
* ELSE
* insert into t3
* (
* mtype,
* mid,
* mname
* )
* values
* (
* m\_type,
* m\_id,
* m\_name
* );
* END IF;
* END
* **createmenu**: This procedure add the foreign key references to the t2 table.
* CREATE DEFINER=`root`@`localhost` PROCEDURE `createmenu`(IN r\_id varchar(45), IN m\_id varchar(45))
* BEGIN
* insert into t2
* (
* rid,
* mid
* )
* values
* (
* r\_id,
* m\_id
* );
* END
* **showdata**: This procedure is to show the data of the restraunts based on the city name.
* CREATE DEFINER=`root`@`localhost` PROCEDURE `showdata`(IN r\_name VARCHAR(45))
* BEGIN
* select \*
* from t1
* where city = r\_name;
* END
* **getmenuitems**: This procedure is to show the Food Item name and Item type for the particular restraunt name entered.
* CREATE DEFINER=`root`@`localhost` PROCEDURE `getmenuitems`(IN g\_name varchar(45))
* BEGIN
* SELECT mname,mtype from t3
* INNER JOIN t2 ON t2.mid = t3.mid
* INNER JOIN t1 ON t2.rid = t1.rid
* where name=g\_name;
* END

**UNIT TESTING**

The unit testiing has been performed using the python **pytest** utility. The pyhton file **test\_testapi.py** must be stored in the same directory as the **ZapApi.py** file.

**Method: Args: Function**

* test\_getrest (<city\_name>, <list(restraunt)>) : Tests the GET method for the restraunt details. Passes test if output and restraunt name match.
* test\_getmenu (<restraunt\_name>,<item\_name>): Tests the GET method for the menu details. Passes test if the output and Item name matches.
* test\_postrest(<restraunt\_id>,<Restraunt\_name>,): Tests the POST method for the restraunt details. Passes test if the output produces successfully added.
* test\_postmenu(<menu\_type>,<iten\_name>,<menu\_id>): Tests the POST method for the menu details. Passes test if the output produces successfully added.
* test\_delmenu(<restraunt\_name>,<restraunt\_name>): Tests the DELETE method for deleting the restraunt details.If the GET call after does not return the restraunt name then test passed.

test\_testapi.py::test\_getrest[Plano-expected0] PASSED

test\_testapi.py::test\_getrest[Garland-expected1] PASSED

test\_testapi.py::test\_getrest[California-expected2] PASSED

test\_testapi.py::test\_getmenu[Wendys-Burger] PASSED

test\_testapi.py::test\_postrest[IPV-Velvet-Pune-Successfully Added] PASSED

test\_testapi.py::test\_postmenu[Lunch-Kadhi-L098-Successfully Added] PASSED

test\_testapi.py::test\_delmenu[Velvet-Velvet] FAILED