Setting up the Sqlite3 db: -

1. Sqlite3 is globally installed in the system using npm i sqlite3 -g.
2. The db is initialized inside the “/server” directory in “db.js” file.
3. The database file is placed in “./server/db” as registrationDB.db

Currently, only user table is present in this file where data related to each user is present there.

The schema for user table is:

[CustomerId] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

[CustomerName] VARCHAR(50) NOT NULL,

[PHONE] VARCHAR(50) NOT NULL UNIQUE,

[EMAIL] VARCHAR(50) NULL,

[ADDRESS] VARCHAR(50) NOT NULL,

[RFID] INTEGER NOT NULL UNIQUE,

[VIN] INTEGER NOT NULL,

[MAC] VARCHAR(50) NOT NULL,

[TOPIC] VARCHAR(50) NOT NULL,

[ACTIVE] INT NOT NULL,

[VEHICLE\_TYPE] VARCHAR(50) NOT NULL,

[FUEL\_TYPE] VARCHAR(50) NOT NULL,

[FUEL\_LEVEL] VARCHAR(50) NOT NULL,

[FUEL\_CAPACITY] VARCHAR(50) NOT NULL,

[MILAGE] DECIMAL(5,2) NOT NULL,

[DateOfCreation] DATETIME NOT NULL,

[DateOfUpdation] DATETIME NOT NULL,

[OtherData] VARCHAR(200) NULL,

CompanyName VARCHAR(50) NULL

1. We can access the registrationDB.db through cmdline using the following command:-

* For opening the sqlite3 database, Inside “./server/db” :-

sqlite3 registrationDB.db

* For adding new user in the db :-

INSERT INTO user(CustomerName,PHONE,EMAIL,ADDRESS,RFID,VIN,MAC,topic,ACTIVE,VEHICLE\_TYPE,FUEL\_TYPE,FUEL\_LEVEL,FUEL\_CAPACITY,MILAGE,DateOfCreation,DateOfUpdation,OtherData,CompanyName) VALUES(‘Raman Raghav’,’9956878412’,’[abc@xyz.in](mailto:abc@xyz.in)’,’21-Camron Street’,32145,36276,’78:87:S6:L0:98’,’car\_4455’,0,’Hatchback’,’disel’,’32%’,’12L’,’60kmph’,’’,’’,’xyz’,’psiborg’);

* Querying entire data(all users) from user table :-

SELECT \* FROM USER;

Equivalent using REST:

*GET /api/*registration

* Query the user using different query string(single & multiple fields can be used) :-

SELECT \* FROM USER WHERE CustomerName=”Raman raghav” AND TOPIC=”car\_4455”;

* + Column values are case-sensitive

Equivalent using REST:

GET /api/registration?name=raman raghav&topic=car\_4455;

* + Field values are case-insensitive

Equivant field names(of api) of different Column name(of database) are mentioned below:

|  |  |
| --- | --- |
| COLUMN NAME | FIELD NAME |
| CustomerId | userId |
| CustomerName | name |
| PHONE | phone |
| EMAIL | mail |
| RFID | rfid |
| ADDRESS | address |
| VIN | vin |
| MAC | mac |
| TOPIC | topic |
| ACTIVE | active |
| VEHICLE\_TYPE | vt |
| FUEL\_TYPE | ft |
| FUEL\_LEVEL | fl |
| FUEL\_CAPACITY | fc |
| MILAGE | milage |
| DateOfCreation | dc |
| DateOfUpdation | du |
| CompanyName | con |

* Delete user by CustomerId from user table :-

DELETE FROM user WHERE CustomerId=0

Equivalent using REST:

DELETE /api/registration/{userId}

* Update user by userId from user table :-

1. Update single field:-

UPDATE user SET ADDRESS=“321 Nehru Street” WHERE CustomerId=0

Equivalent using REST:

PUT /api/registration/{userId}

* + Request body has following structure:

{“ADDRESS”:”321 Nehru Street”}

2. Update multiple fields:-

UPDATE user SET ADDRESS=“321 Nehru Street”, VEHICLE\_TYPE=”SUV”, FUEL\_TYPE=”disel”, MILAGE=”120kmph” WHERE CustomerId=0

Equivalent using REST:

PUT /api/registration/0

* + Request body has following structure:

*{*

*“ADDRESS”:”*321 Nehru Street*”,*

*“VEHICLE\_TYPE”:”SUV”,*

*“FUEL\_TYPE”:”disel”,*

*“MILAGE”:”120kmph”*

*}*