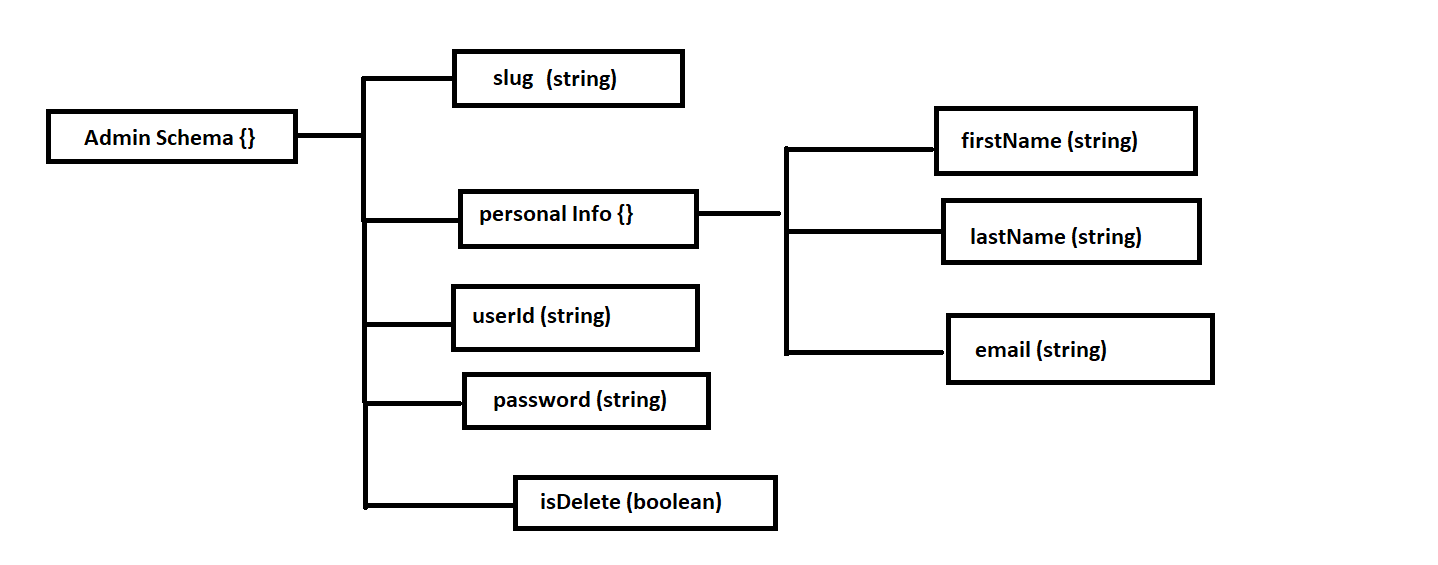
**API Documentation**

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **SL** | **Schema Name** | **Page Number** |
| 01 | Admin | 3 |
| 02 | Client | 4 - 7 |
| 03 | User | 8 – 11 |
| 04 | Blog |  |
| 05 | Official | 21 - 26 |
| 06 | Comment | 27 – 28 |
| 06 | Contact | 29 - 31 |

**Admin Schema**



**1. Create a new Admin**

**AP**I: baseUrl/admin eg: http://localhost:3030/admin

**Type: GraphQl**

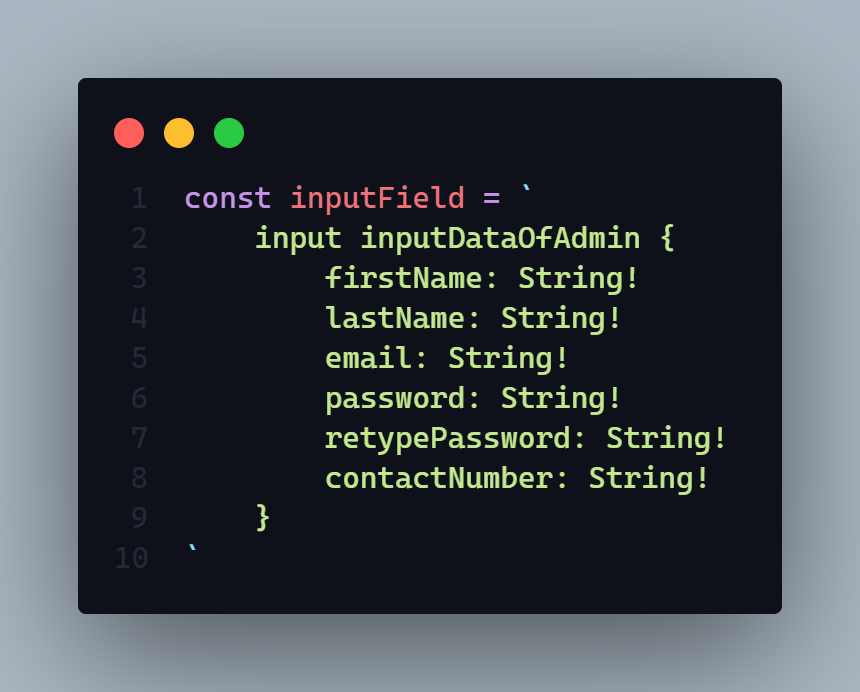
**Method :** POST / Mutation

**Access**: **:** All

**Body :**

**Another Data =>**

1. contactNumber**: String** (required)
2. firstName **L String** (required)

****

**Params:** N/A

**Query:** N/A

**Response:**



Here

* **message** will sent a response message. Every time it will come mandatory
* **data** will sent create admin data
* **status** will return a http status code here it will sent **406** and **201**

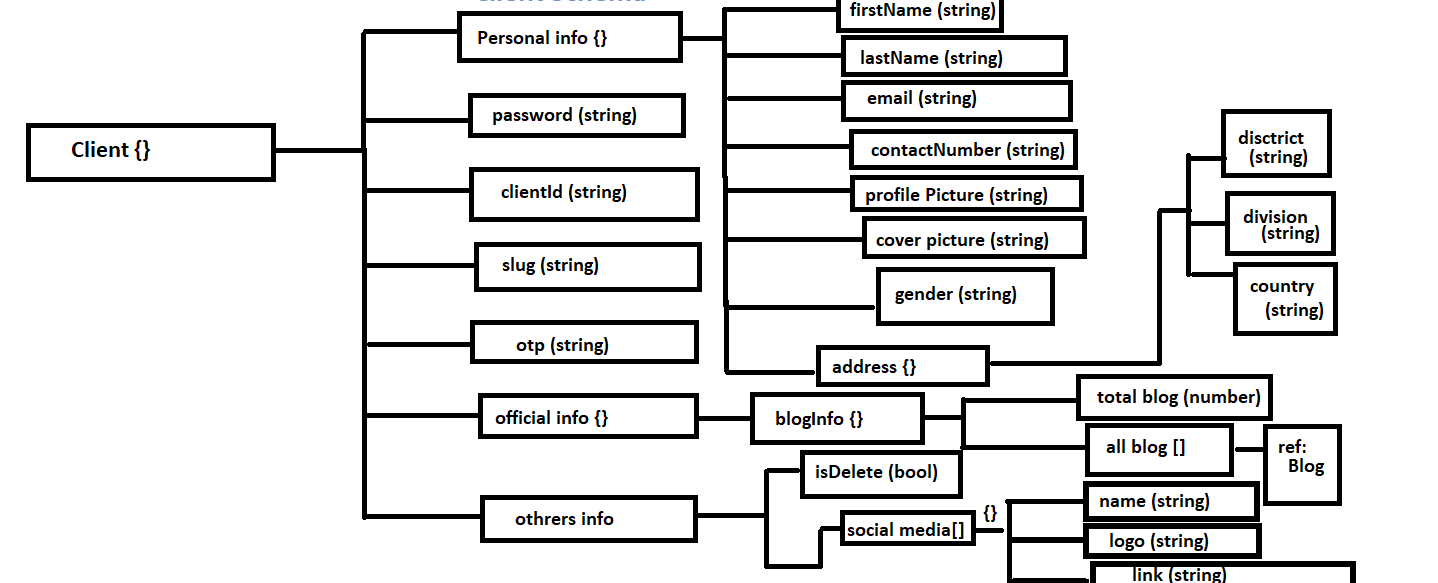
**Demo:**

****

**Description:** If status code is

* **201 =>**
  + Admin successfully created
* **406 =>** 
  + Any type of runtime error
  + Data Joi validation error
  + User Creation Failed
  + Admin Creation Failed
  + Admin Id is not unique
  + Password Hashing Problem
  + Email is exist please try with another email

**Client Schema**



**1. Create a new Client**

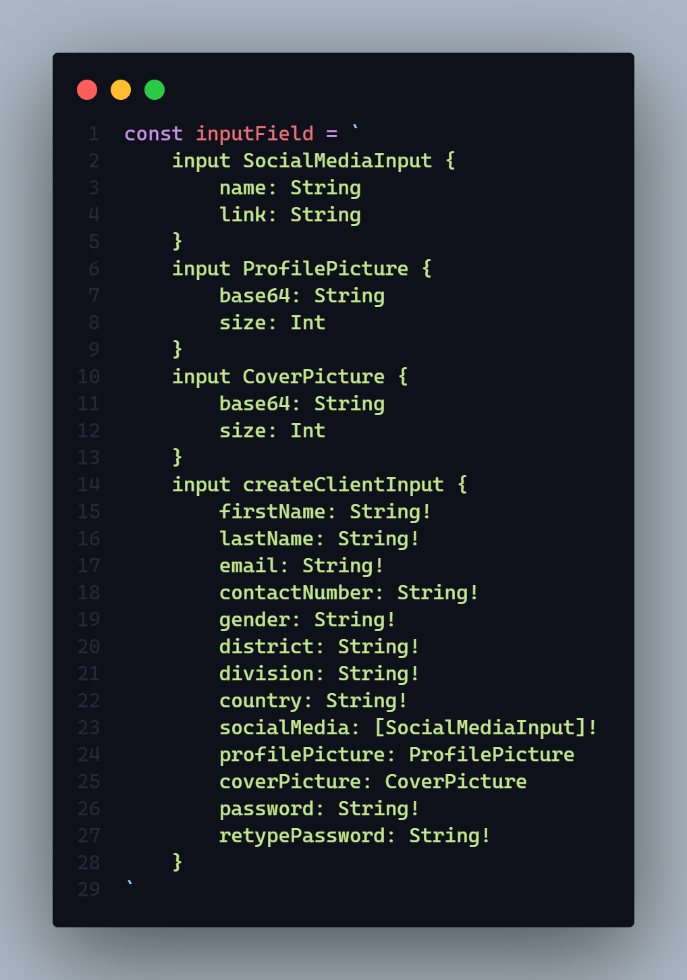
**AP**I: baseUrl/client eg: http://localhost:3030/client

**Type: GraphQl**

**Method :** POST / MUTATION

**Access**: **:** client

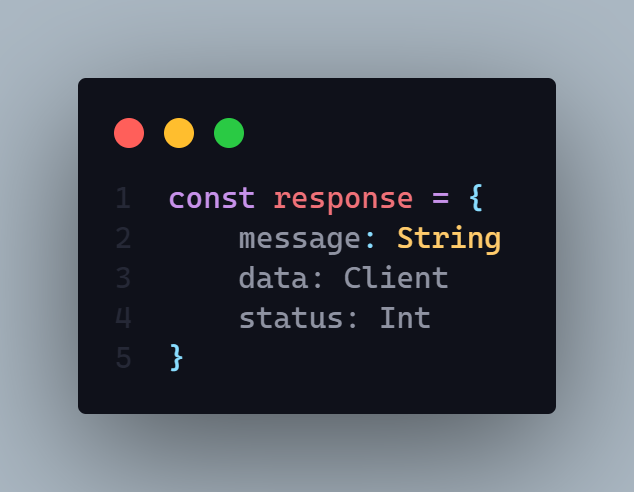
**Body :**



**Params:** N/A

**Query:** N/A

**Response:**

****

**Description:** If status code is

* **201 =>**
  + Client successfully created
* **406 =>** 
  + Any type of runtime error
  + Data Joi validation error
  + User Creation Failed
  + Client Creation Failed
  + Password Hashing Problem
  + Email is exist please try with another email
  + Only Jpg Jpeg Png are allowed
  + Cover picture upload failed
  + Client id is not unique
  + Profile Image upload failed

**Demo:**

****

**2. Delete a client by slug**

**AP**I: baseUrl/client eg: http://localhost:3030/client

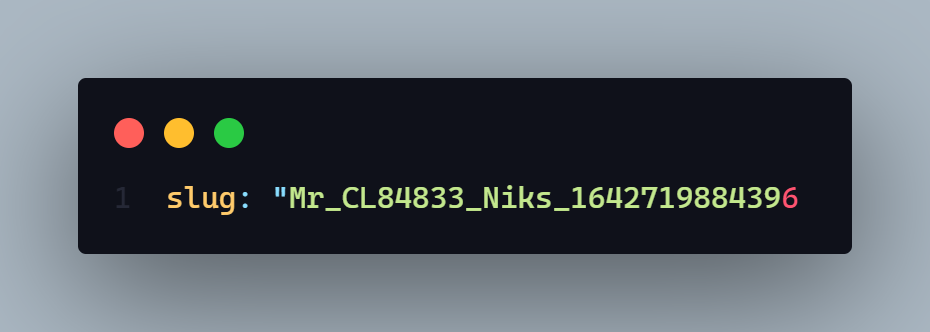
**Type: GraphQl**

**Method :**POST / MUTATION

**Access**: **:** admin, client

**Auth : true**

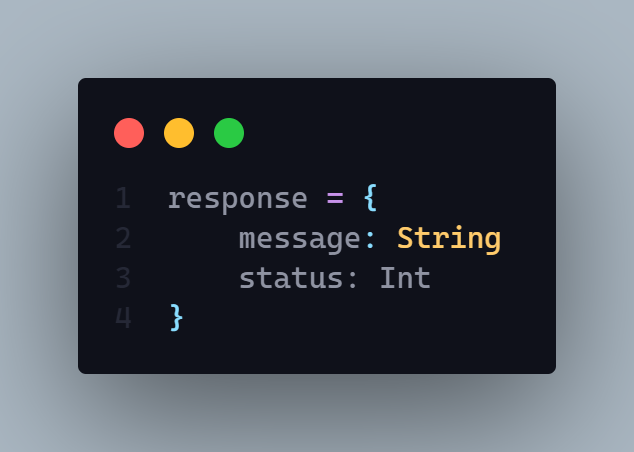
**Body :**



**Params:** N/A

**Query:** N/A

**Response:**

****

**Description:** If status code is

* **202 =>**
  + Client has been successfully deleted
* **406 =>** 
  + Any type of runtime error
* **304 =>**
  + Client delete failed
* **401** =>
  + Permission denied
  + Unauthorized user

As a input we need only the **slug** of respected user.

**Demo:**



**3. Client can see only his activity by year.**

**AP**I: localhost:3030/client/activity

**Type: REST**

**Method :** GET

**Access**: **:** Client

**Auth : true**

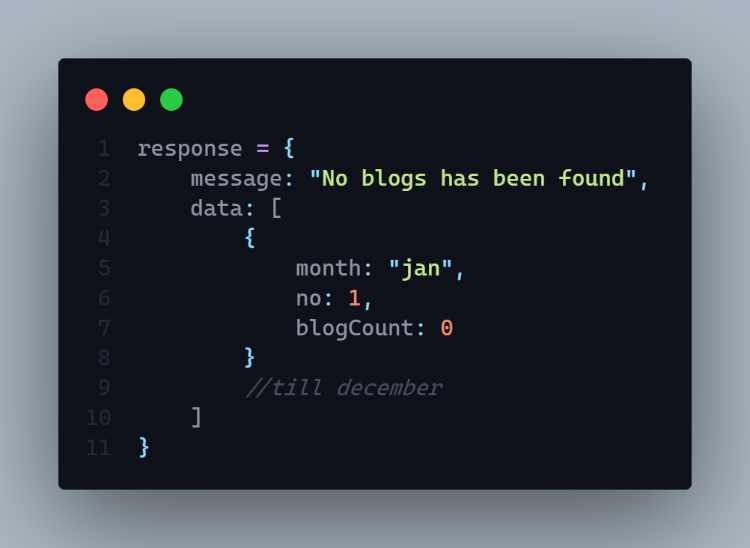
**Body :  N/A**

**Params:** N/A

**Query:**

* year : String eg: localhost:3030/client/activity?year=2021

**Response:**

****

**Description:** If status code is

* **202 =>**
  + Amount of blog has been found
  + No blogs has been found
* **200 =>** 
  + Any type of runtime error
  + Something went wrong

**Hints:** It will query how many blog logged in user published within a year by month

**4. Update client data by id**

**AP**I: localhost:3030/client

**Type: GraphQL**

**Method :** POST / MUTATION

**Access**: **:** admin, client

**Auth : true**

**Body :**



**Params:** N/A

**Query:** N/A

**Data:** Full name, email, password, contact number, bio, location, socialMedia

**Hints:** N/A

**Response:**

****

**Description:** If status code is

* **202 =>**
  + Client updated successfully
* **406 =>** 
  + Any type of runtime error
* **304 =>**
  + Client update failed
* **401** =>
  + Permission denied
  + Unauthorized user
* **404 =>** 
  + Slug required

As a input we need only the **slug** of respected user. And also updated data input.

**Demo**

****

**5. Client can see only his published or draft blog**

**AP**I: localhost:3030/client

**Type: GraphQl**

**Method :** Mutation / POST

**Access**: **:** “client”

**Auth : true**

**Input Body :**

1. **sortBy:** String
2. **pageNo:** Number
3. **limit:** Number
4. **searchFor :** String (value will be “draft” or “publish” )



**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints: Sort by –**

1. Latest
2. A – Z (by title name)
3. Z- A (by title name)
4. Viewers (high to low )

**Pagination Include**

**Data limit include (**Data will return by limit dynamicaly**)**

**Data Query Searching Value will be :** draft or publish

**Response:**

****

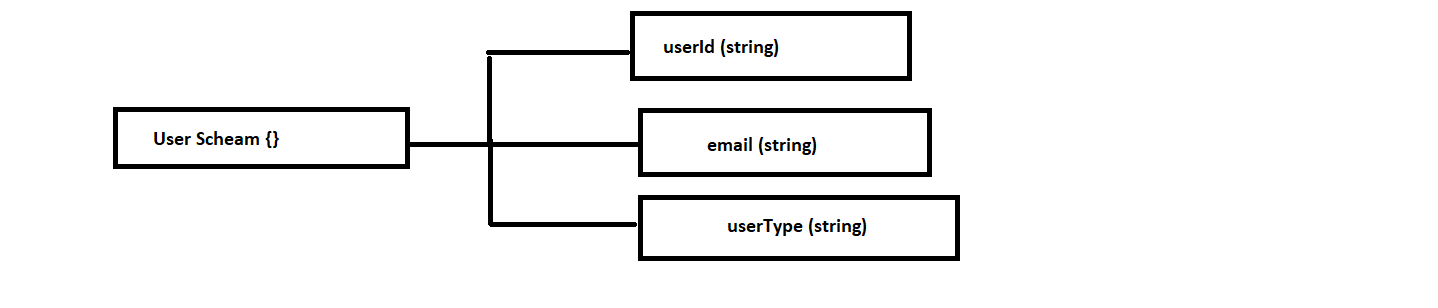
**Description:** If status code is

* **202 =>**
  + X blog found (x = any number (0 – 9 ))
* **406 =>** 
  + Any type of runtime error
* **401** =>
  + Permission denied
  + Unauthorized user
* **404 =>** 
  + Blog not found

**Demo :**



**User Schema**



**1. Upload or update logged in user profile picture or cover picture and delete existing one from server.**

**AP**I: localhost:3030/user

**Type: GraphQL**

**Method :** POST / MUTATION

**Access**: **:** admin, client

**Auth : true**

**Input Body :**

1. **uploadType** => It will be string. (required)
2. **base64** => it will be a string (required)
3. **size** => It will be number (required)



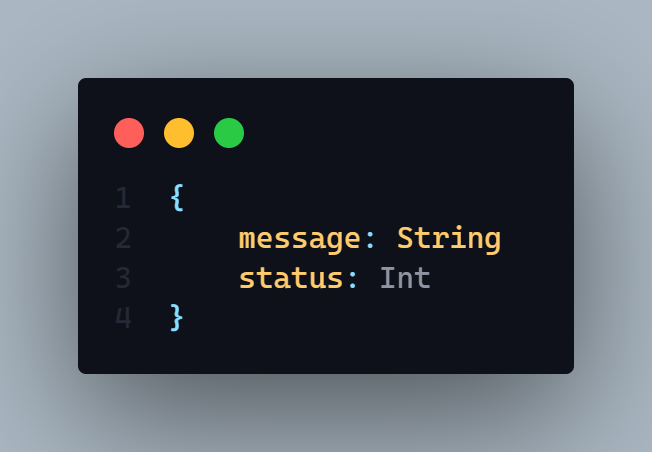
**Params:** N/A

**Query:** N/A

**Hints:** N/A

**Response:**

1. **messege** => It will be string. (required)
2. **status** => it will be a string (required)

****

**Description:** If status code is

* **205 =>**
  + This controller is inComplete for admin
  + Upload Type Required
  + Existing file failed to delete
  + Something is wrong please try again later
  + Only Jpg Jpeg etc are accepted
  + Image Upload Failed
* **406 =>** 
  + Image upload successfully
* **401** =>
  + Permission denied
  + Unauthorized user
* **304 =>** 
  + User image update  failed
* 402 =>
  + This controller is inComplete for admin

Demo:



**2. User can see his profile**

**AP**I: localhost:3030/user/profile

**Type: GraphQL**

**Method :** GET

**Access**: **:** All

**Auth : true**

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** It just take the token from valid cookies then give the expected token to the backend.

**Response:**

**Description: :** If status code is

* **202 =>**
  + message: "User found",

**3. Login user with email and password**

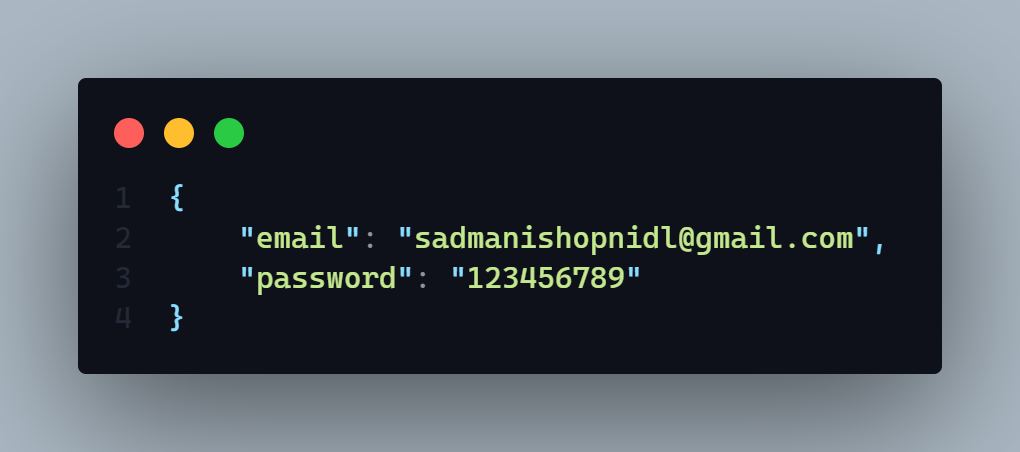
**AP**I: localhost:3030/user/login

**Type: REST**

**Method :** POST

**Access**: **:** ALL

**Body :**



**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** Logged in user by email and password after match the data create a token and set is as auth in the browser cookies.

**Response:**

****

**Description:** If status code is

* **202 =>** Login Successful
* **200 =>** 
  + Token Creation error
  + Password does not match
  + User Not found

It will set the logged in jwt token into cookies by using cookie – parser.

Cookie name is => **auth**

**4. Rewrite a logged in user password and verify it**

**AP**I: localhost:3030/client/verify

**Type: REST**

**Method :** POST

**Access**: **:** admin, client

**Auth : true**

**Body :**

1. Password => String

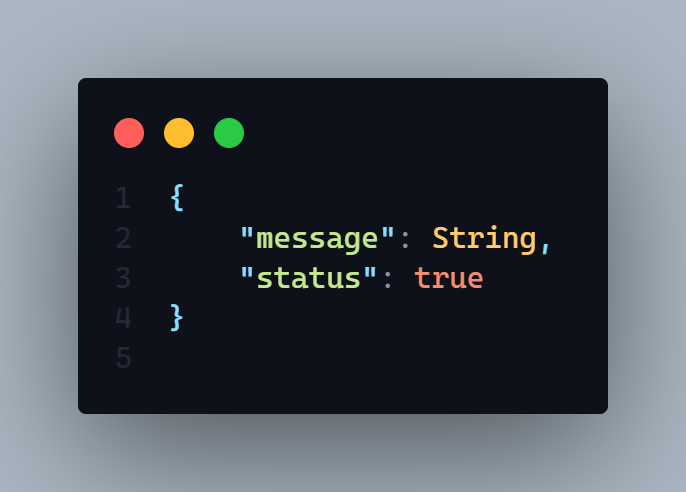


**Params:** N/A

**Query:** N/A

**Data:** N/A

**Response:**

****

**Description:** If status code is

* **202 =>** Password Matched
* **200 =>**
  + User not found
  + Password mismatch
  + Database password not found
  + Any run time error

**Hints:** It will take a input of password and match it with the data base password of the logged in user

**5. Update logged in user password. (rest)**

**AP**I: localhost:3030/user/update/password

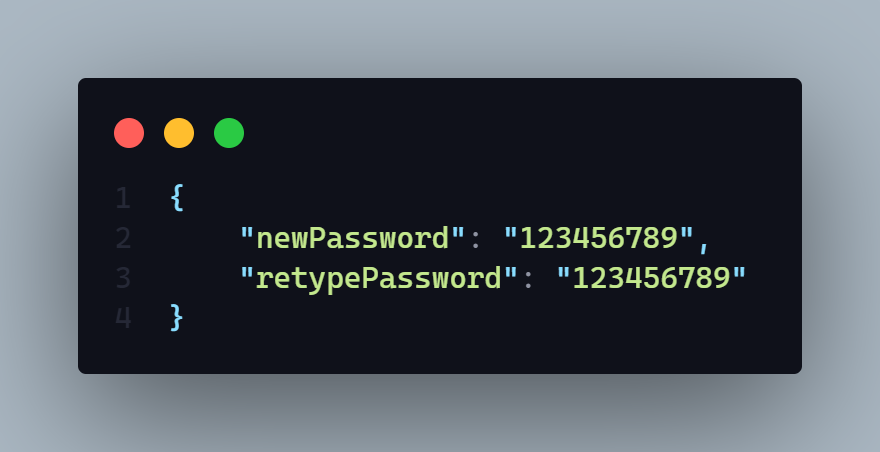
**Type: REST**

**Method :** PUT

**Access**: **:** admin, client

**Auth: true**

**Body :**

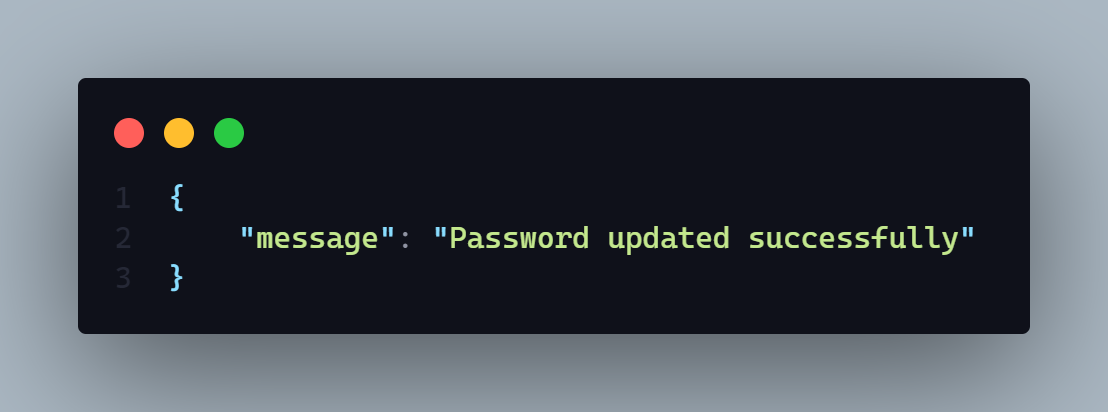


**Params:** N/A

**Query:** N/A

**Data:** N/A

**Response:**

****

**Description:** If status code is

* **202 =>** Password updated successfully
* **200 =>**
  + User not found
  + Password update failed
  + Password hashing problem
  + Any run time error

**Hints:** It will take newPassword and retypePassword as an input then it will match both password and if it will update the logged in user database password

**5. forgot Password part - 1. Take the registered phone number and sent an OTP in the phone as a text message (rest)**

**AP**I: localhost:3030/user/forgot/password

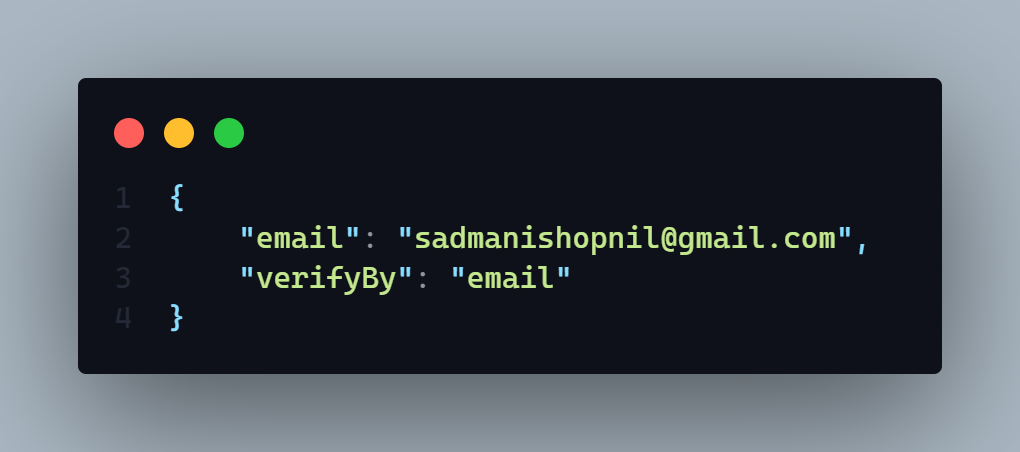
**Type: REST**

**Method :** POST

**Access**: **:** N/A

**Auth: false**

**Body :**



**Params:** N/A

**Query:** N/A

**Hints:** It will set a jwt token as a response in the cookie section name **verifyToken.** And also it will sent a new otp to the respected user mobile if user select verify via mobile or in a email if user select verify by email

**Response:**

****

**Description:** If status code is

* **202 =>** Password updated successfully
* **200 =>**
  + User not found
  + OTP not save
  + Email sent failed
  + Any run time error
  + OTP message not send
  + Somethings went wrong
  + OTP creation failed
  + User not found try with another mail

**6. Forgot Password part - 2. Take the OTP as a input and match with the database one if match it will response back as a boolean.**

**AP**I: localhost:3030/user/verify/otp

**Type: REST**

**Method :** POST

**Access**: **:** N/A

**Auth: false**

**Body :**

1. otp: String



**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** It will take an OTP and verified it with database otp and after verified it will delete exist database otp of that particular user and response back a positive boolean.

**Response:**

****

**Description:** If status code is

* **202 =>** OTP successfully verified
* **200 =>**
  + OTP not verified please put a valid one
  + OTP required or not valid
  + Email sent failed
  + Any run time error
  + User not found

7. **Forgot Password part - 3. Last part it will take new password input and update the password.**

**AP**I: localhost:3030/user/reset/password

**Type: REST**

**Method :** POST

**Access**: **:** ALL

**Body :**



**Params:** N/A

**Auth: false**

**Query:** N/A

**Data:** N/A

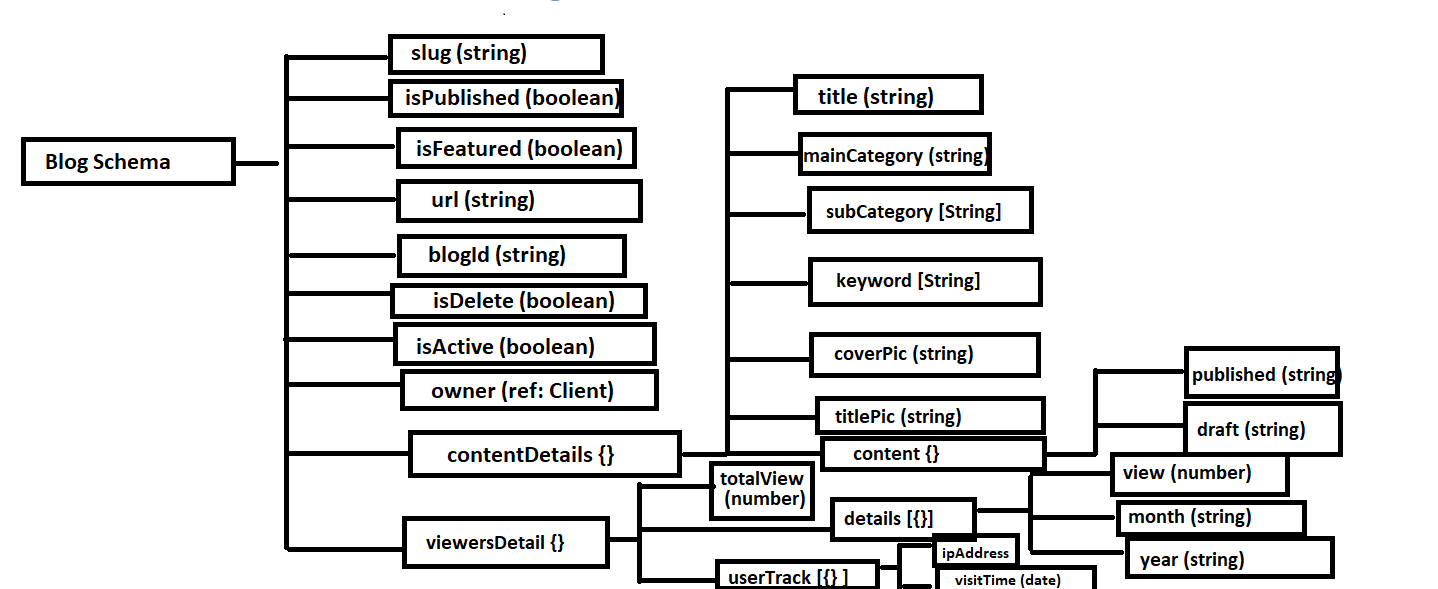
**Hints:** It will take the new password with retype password and change it

**Response: It only response a new message in an object**

**Description:** If status code is

* **202 =>** Password has changed successfully
* **200 =>**
  + Password hashing problem
  + Password change failed
  + Any run time error

**Blog**  **Schema**

****

**1. Get blog by --**

**Query by –**

* **Recent Blog**
* **Top of this month** 
  + **Query by viewer count of current month**
* **Most read blog**
  + **By viewer count all time**
* **All featured blog**
* **By main category**
* **If no query then show all blog**

**Extra Fetcher –**

* **Search 🡺** 
  + **By main category**
  + **By Sub Category**
  + **By Title**
  + **By keyword**
    - **Search input will have to perfectly match**
* **Pagination Include**
* **Filter by =>** 
  + **Sub Category**
  + **Publish Year**
* **Data limit will provide dynamically If not provide then will give all found data**
* **Sorted by –**
  + **Latest**
  + **A – Z** 
    - **By title name in Ascending order**
  + **Z – A** 
    - **By title name in descending order**
  + **View** 
    - **In Descending Order**

**AP**I: baseUrl/blog eg : localhost:3030/blog

**Type: GraphQL**

**Method :** POST / Query

**Access**: **:** ALL

**Auth: false**

**Body :**

**Query Part**

1. If User want to get **Recent Blog =>** 
   1. queryBy : String [expected Data => **“recent”** ]
2. If User want to get **Top of this month =>** 
   1. queryBy : String [expected Data => **“topMonth”** ]
3. If User want to get **Most Read blog =>** 
   1. queryBy : String [expected Data => **“mostRead”** ]
4. If User want to get all **Featured Blog =>** 
   1. queryBy : String [expected Data => **“featured”** ]
5. If User want to get blog by  **Main Category =>** 
   1. queryBy : String [expected Data => **“mainCategory”** ]
   2. queryInput: String [expected Data => “”] **=> this expected data will be any main category of blogs.**

**Filtering Part**

1. If User want to filter blog by  **Published year =>** 
   1. filter By : {

publishedYear: String [expected data => “2022” or “2021” …..]

}

1. If User want to filter blog by  **Sub Category =>** 
   1. filter By : {

subCategory : [String] [expected data => [“web Development”, “web design”] ]

}

**Search Part**

1. If User want to search blog **=>** 
   1. **search : String** [expected input will be a string]

**Sorting Part**

1. If user want to sort blog =>
   1. sortBy : String [Expected input will be a string]
      1. option => 1
         1. latest
         2. A\_Z
         3. Z\_A
         4. View

**Page No:**

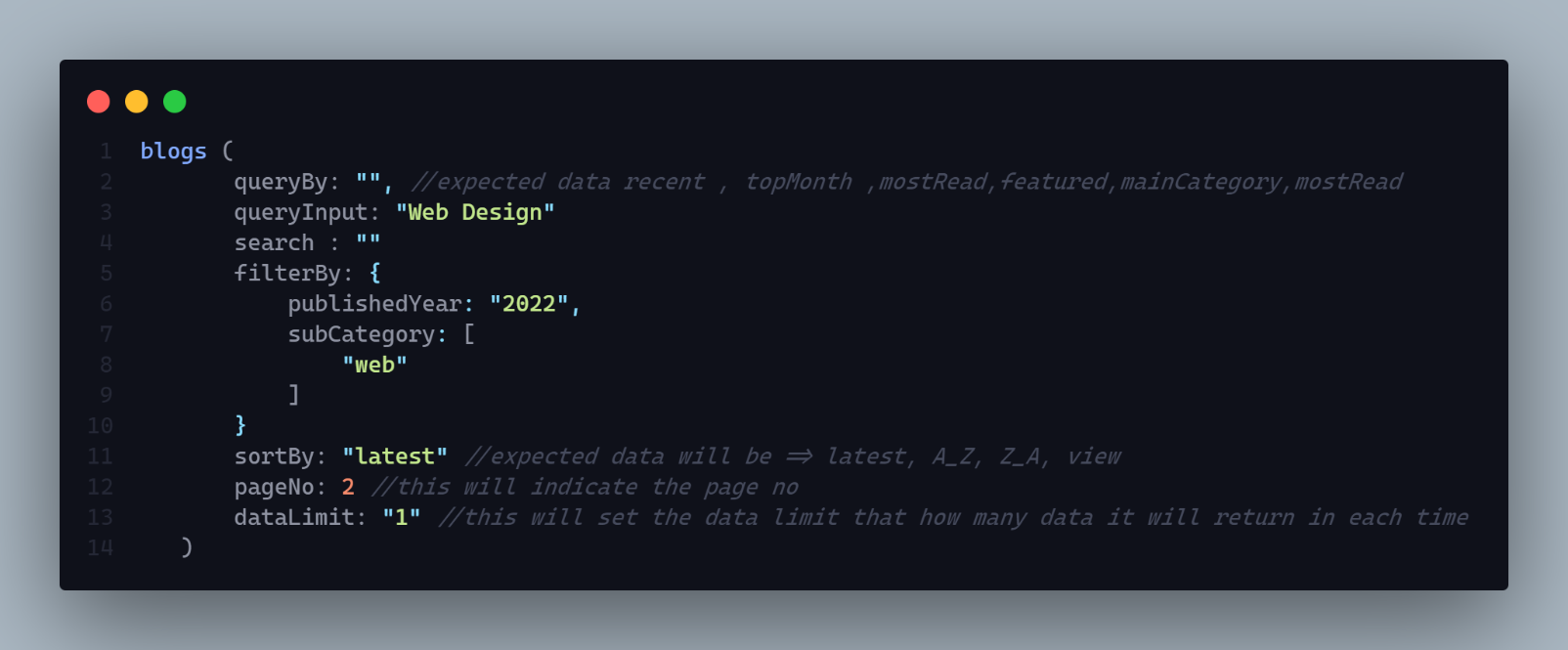
1. **If** user want to give page number = >
   1. **pageNo: Int [**expected input is a integer number like 1,2,3**]**

**Data Limit**

1. **If user want to declare that how many data will need to show in a single page then it need to be declare by this way** 
   1. **dataLimit: String [** Expected input will be a string of integer number like “5” **] it means server will response back 5 data each page.** 
      1. **By default if user don’t give any data limit then it will show 5 data**

**each time.**

**If User don’t give any attribute then it will return all blogs**

**Body Demo:**

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

****

**Description:** If status code is

* **202 => n** number blog found
* **404 => No blog found**
* **406 => Any runtime error**

**Work Of this API:**

1. **This api will query by many ways like (**recent blog, topMonth.. etc**) and give expected blog to user**
2. **User also can search blog by this api**
3. **User can also filter by many ways**
4. **User also can get data by page. Pagination included.**
5. **User also can sort data many way like sort by => (**latest, view… etc**)**

**Demo:**



**2. Get Two Top of the month blog category by viewers and then get single top blog from each category of the current month.**

**AP**I: localhost:3030/blog

**Type: GraphQL**

**Method :** POST / Query

**Access**: **:  all**

**Auth: false**

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** It will query top two blog category and return a single blog data from each blog

**Response:**

****

**Description:** If status code is

* **202 => Blog found**
* **404 => No blog found**
* **406 => Any runtime error**

**WorkFlow :**

1. This Api will give two top category of the month
   1. This is query by how many visitors visit a blog in a single month. And got the top two blog by visitors and find the main category of that blog
2. This api will give a single blog information from each of two top category of the month.

**Demo**

****

**3. Mark Blog as featured**

**AP**I: localhost:3030/blog

**Type: GraphQL**

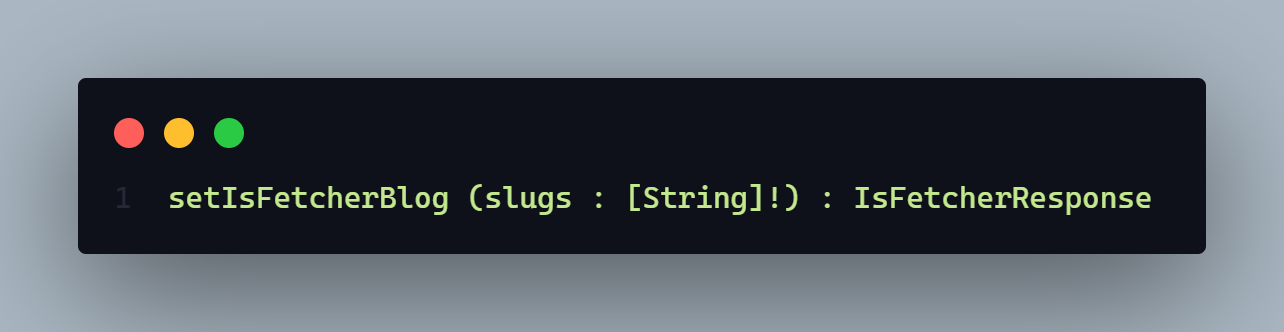
**Method :** POST / MUTATIOn

**Access**: **:** admin

**Auth: true**

**Body :  Need to pass =>**

1. **slugs (paramenter name ) ==🡺 array [] (parameter data type)**

****

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** This api can make multiple blog as a fetcher one.

**Response:**

****

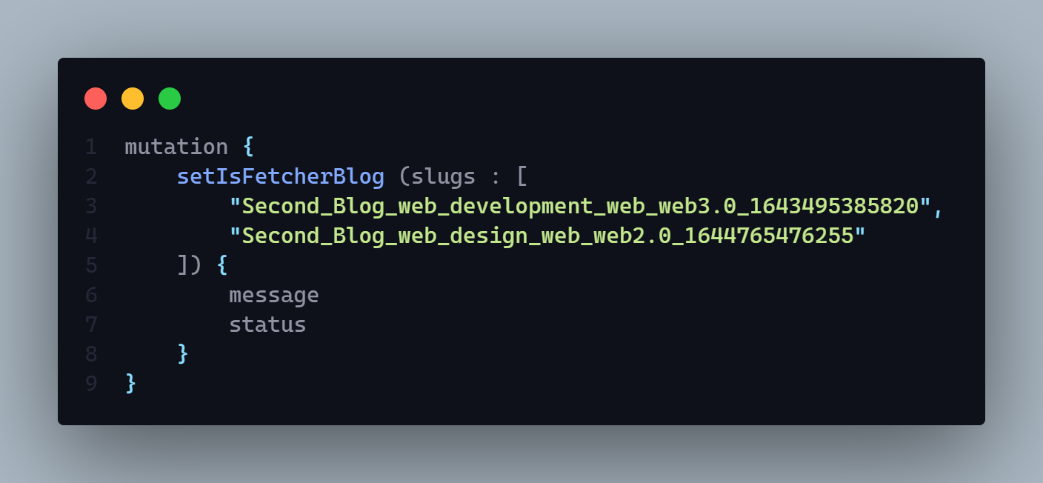
**Description: :** If status code is

* **202 => Blog has successfully featured**
* **304 => Blog featured failed**
* **406 => Any runtime error**
* **401 => Permission denied**

**Work Flow :**

1. This Api will make multiple blog as a featured one.
2. We need to give slug of blog in an array for make multiple featured blog at a same time.
3. Only Admin can access this api
4. Authentication requried

**DEMO** :



**4. Count a new view if a user click on individual blog page and stay their at least 60 sec**

**AP**I: localhost:3030/blog

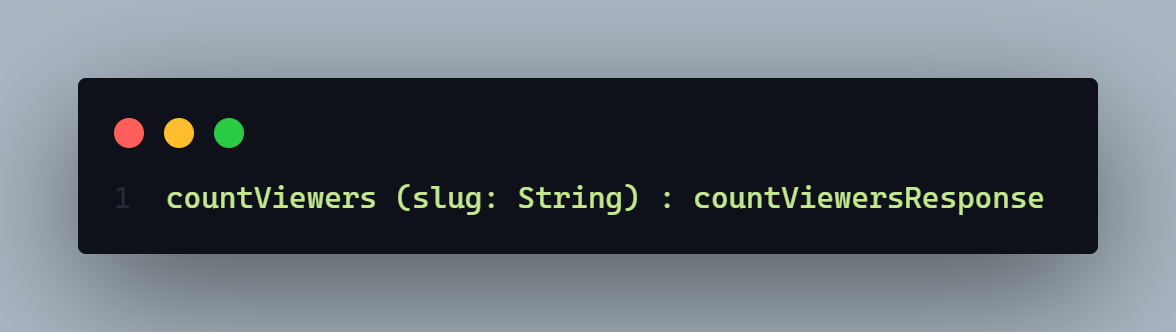
**Type: GraphQL**

**Method :** POST / MUTATION

**Access**: **:** ALL

**Auth** : False

**Body :   We** need to pass a blog slug name. By this slug we can count a new viewers of that particular blog



**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

****

**Description:** If status code is

* **202 => View count!!**
* **304 => Blog featured failed**
* **406 => Any runtime error**
  + **View count failed**
* **403 => View count failed due to limit exceed**
  + **View count failed due to own blog**
* **404 => Ip Address requeried**

**Work Flow :**

1. From this Api we can count a blog viewers with some conditions.
2. Conditions =>
   1. Logged in user can not increase own blog view
   2. View will be count of a user only one time in a day by tracking the user device ip address.
   3. It will increase the total view also and keep tack the user view details including the user ipAddress.

Demo:



**5. Get the all amount of blog available of all categories in the website or official schema**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**6. Get all sub category name available from what blog categories blog we want to get .**

**AP**I: localhost:3030/blog

**Type: GrapgQL**

**Method :** POST / Query

**Access**: **:** N/A

**Is Auth :** False

**Body :** In the body it needs to pass a single parameter =>

1. category => value will be the what main category blog we want to query



**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** It will first get all blog data by respected blog category and get all unique subcategory it has found from all blogs.

**Attributes = >  
 1. Input body :**

**a.** category => blog Main Cateogory \*

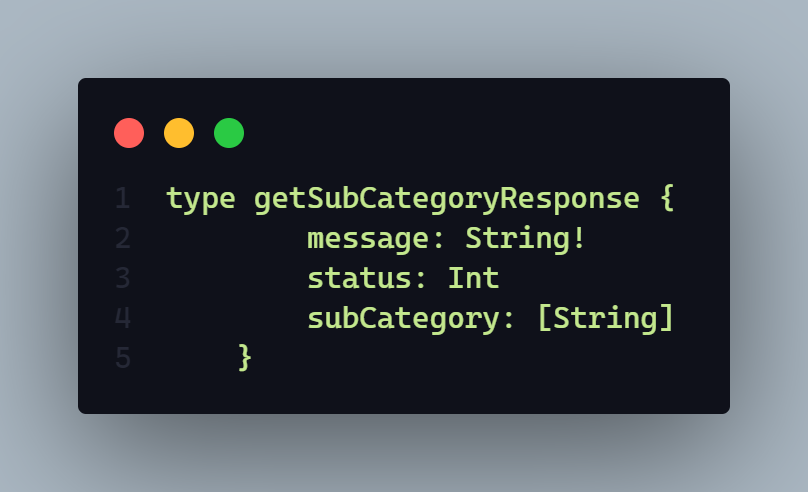
**2. Output response:**

**a.** message => It will send response back a message (\*)

b. status => it will send response a http status code (\*)

c subCategory => it will send response all subCategory in a array.

**Response:**

****

**Description:** If status code is

* **202 => Subcategory found!!**
* **406 => Any runtime error**
* **404 => No subcategory found!!**

**Work Flow :**

1. When user want to see blog by main category it will show all sub categories of founded blog’s
2. It will First query all blog and then find the subcategory from all founded blog.

**Demo :**

****

**NB: (\*) means mandatory attribute**

**7. Get Individual published blog by slug with related blogs also --**

* **Related blog will be query by** 
  + Select blog category
  + Select Blog sub category
  + Keyword

**AP**I: localhost:3030/blog

**Type: GraphQL**

**Method :** POST / QUERY

**Access**: **:** ALL

**Body :** In the body it needs to pass a single parameter =>

1. slug => this is the slug of a blog what’s value we want to get.



**Params:** N/A

**Query:** N/A

**Attributes = >  
 1. Input body :**

**a.** slug => this will be a string type data what represent the blog we want to get the value\*

**2. Output response:**

**a.** message => It will send response back a message (\*)

b. status => it will send response a http status code (\*)

c. blog => it will send a response of query blog.

d. relatedBlog => it will send a response all related blogs of query blog. It will return a array of blog in descending order by blog title name.

**NB: (\*) means mandatory attribute**

**Data:** N/A

**Hints:** It will query a blog by blog slug and give that blog to us. It also give all related blogs of that particular blog.

**Response:**

****

**N.B:** Here **message** and **status** fields are mandatory

**Description:** If status code is

* **202 => Blog found!!**
* **406 => Any runtime error**
* **404 => Blog not found!!**

**Work Flow :**

1. This api will give a single blog all data.
2. It is query by slug
3. With a single blog response it also sent a response of all related blogs of that particular blog
4. All related blog are query by
   1. Blog main category
   2. Blog keyword
   3. Blog sub category
5. All related blog query action in or process
6. It only show published and active blog

**Demo :**

****

**8. Save Blog**

* **If user creates new blog then create new one and store data in temporary space**
* **If User updates and save, then only store the new data into existing one that time we should have blog id to find that blog**
* **During update time it will return then updated blog also**

**AP**I: localhost:3030/blog

**Type: GraphQL**

**Method :** POST

**Access**: **:** client

**Body :**



**Params:** N/A

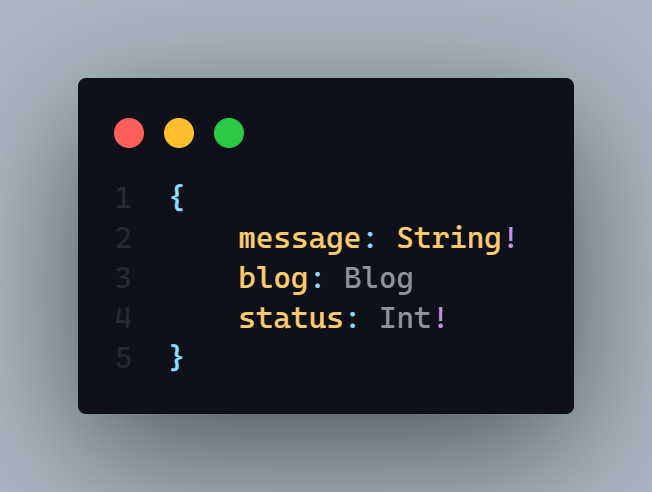
**Query:** N/A

**Auth: true**

**Data:** N/A

**Hints:** This will save blog content into draft section. If user save the blog in the first time then it will create a new blog and then save it. But if blog is exist then user find the blog by blogId and then update the content into draft section.

**Response:**

****

**Description:** If status code is

* **201 =>** Blog save successfully
* **406 =>**
  + Any run time error
  + Blog is not save
* **401** =>
  + Permission deniend
  + Unauthorized user

**In the time of Save blog 1st time =>**

**Send Data =>**

1. **Title , mainCategory content is mandatory.**

**In the time of update or save a exist blog =>**

**Send Data =>**   
 **1.blogId is mandatory**

**Demo:**



**(\*\*) means mandatory data during save a fully new blog. But if blog exist then only blogId and content part is mandetory**

**9. Published a blog**

* **If user creates and direct publish then blog will be create and publish and store content both in temporary or draft and published section**
* **If user published later in the draft section then only update the published section.**

**AP**I: localhost:3030/blog

**Type: GraphQL**

**Method :** POST

**Access**: **:** client

**Auth: true**

**Body :**



**Here blogId is mandatory when it will be a updating phase I mean when blog is already created. And input part is needed when publish a blog directly**

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** If user direct publish a blog that time input filed data need to given to create a new blog. But if user publish blog later then just need to provide the blogId so that respective blog can be find by blogId.

**Response:**

****

**Description:** If status code is

* **202 =>** Blog published successfully
* **406 =>**
  + Any run time error
  + Blog published failed
* **401** =>
  + Permission deniend
  + Unauthorized user
* **404** =>
  + Blog not found

**10. Preview a blog**

* **It generally give individual blog by blog id** 
  + **It will query blog content from draft section**

**AP**I: localhost:3030/blog

**Type: GraphQL**

**Method :** POST

**Access**: **:** client

**Auth: true**

**Body :**

1. blogID => by this blog id blog will be query and show. It is mandatory



**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** This api will query blog by blogId and from this query client will get the draft section of blog

**Response:**

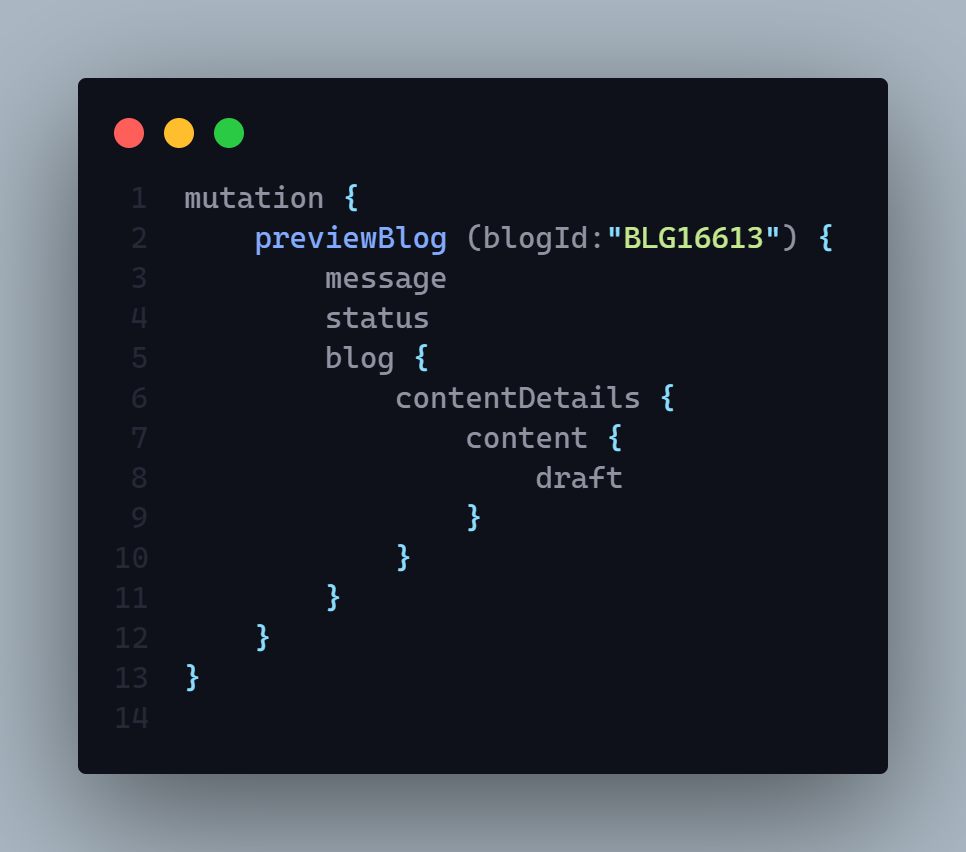
****

**Description:** If status code is

* **202 =>** Blog found
* **406 =>**
  + Any run time error
* **401** =>
  + Permission deniend
  + Unauthorized user
* **404** =>
  + Blog not found

**This api will give a particular blog’s draft section as a preview of that blog**

**Demo:**



**11. Delete an individual blog by slug**

**AP**I: localhost:3030/blog

**Type: GraphQL**

**Method :** POST

**Access**: **:** client, admin

**Auth: true**

**Body :**



**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

****

**Description:** If status code is

* **202 =>** Blog deleted successfully
* **406 =>**
  + Any run time error
* **401** =>
  + Permission deniend
  + Unauthorized user
* **406** =>
  + Blog deleted failed

**Demo**

****

**12. Update blog element by slug**

**1. sub category**

**2. Main Category**

**3. Keyword**

**AP**I: localhost: 3030/blog

**Type: GraphQL**

**Method :** POST

**Access**: **:** client

**Body :**

****

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

****

**Description:** If status code is

* **202 =>** Blog updated successfully
* **406 =>**
  + Any run time error
  + Blog update failed
* **401** =>
  + Permission deniend
  + Unauthorized user
* **404** =>
  + Blog not found

**Demo**



**13. Update blog title or cover picture by slug**

**After update you have to delete the previous one from database and server**

**AP**I: localhost:3030/blog

**Type: GraphQL**

**Method :** POST

**Access**: **:** client

**Body :**

1. Input =>
   1. base64 => **String** (mandatory)
   2. size => **Int** (mandatory)
2. slug => **String** (mandatory)
3. type => **String** (mandatory)
   1. value == “**cover**” for update cover picture
   2. value == “**title**” for update title picture



**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** It will update cover and title picture both. But if user input type = cover then it will update the cover picture and If user input type = title then it will update the title picture and also delete the existing file from server

**Response:**

****

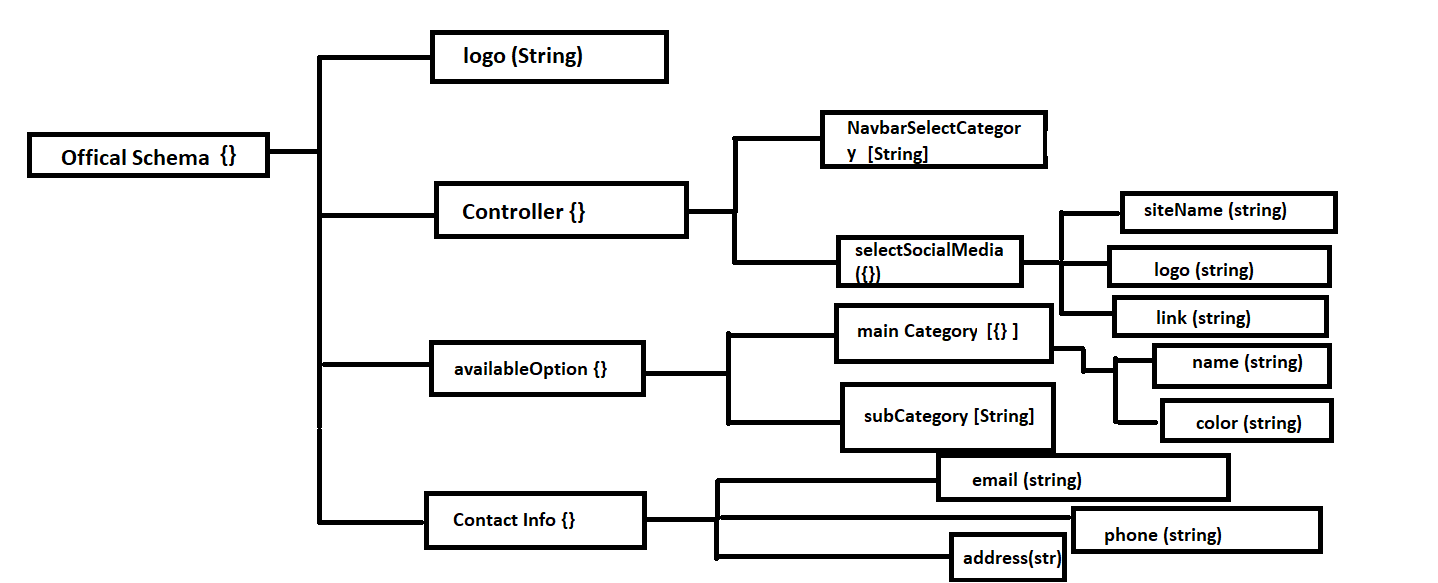
**Description:** If status code is

* **202 =>** 
  + Cover image successfully updated
  + Blog titile image successfully updated
* **406 =>**
  + Any run time error
  + Cover image update failed
  + Blog title image update failed
  + Only allow jpg png and jpeg format
  + Cover picture upload failed
  + Blog Title picture upload failed
  + Previous image delete failed
* **401** =>
  + Permission deniend
  + Unauthorized user

**Demo**

****

**Official**  **Schema**

****

**1. Create an official schema**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**2. Get all Official schema’s every info**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**3. Create available main category with color**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**4. Create available sub category**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**5. Get all main category in ascending order**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**6. Get all sub category in ascending order**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**7. Add contact social media to contact officially**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**8. Update contact social media to contact officially**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**9. Delete contact social media by site name**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**10.** **Select main category fo sr navbar (gql)**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

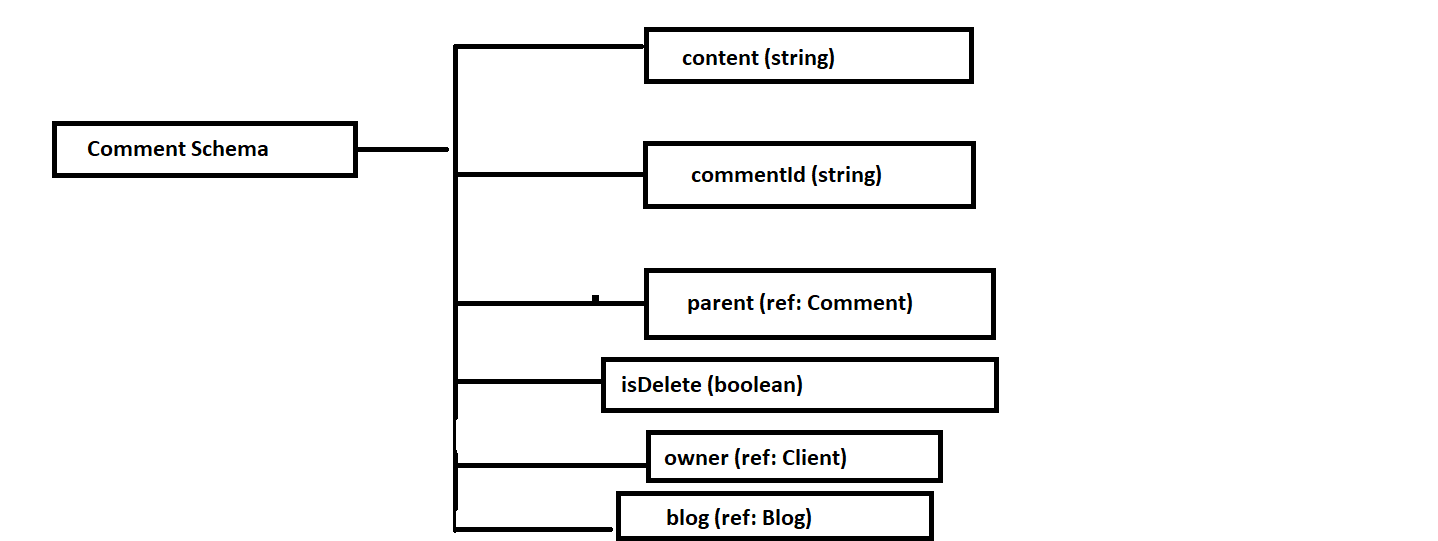
**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**Comment**  **Schema**

****

**1. Create a new comment or reply in the time of reply you need to pass parent comment id**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**2. Update comment by comment id**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**3. Get all comment by blog id name**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

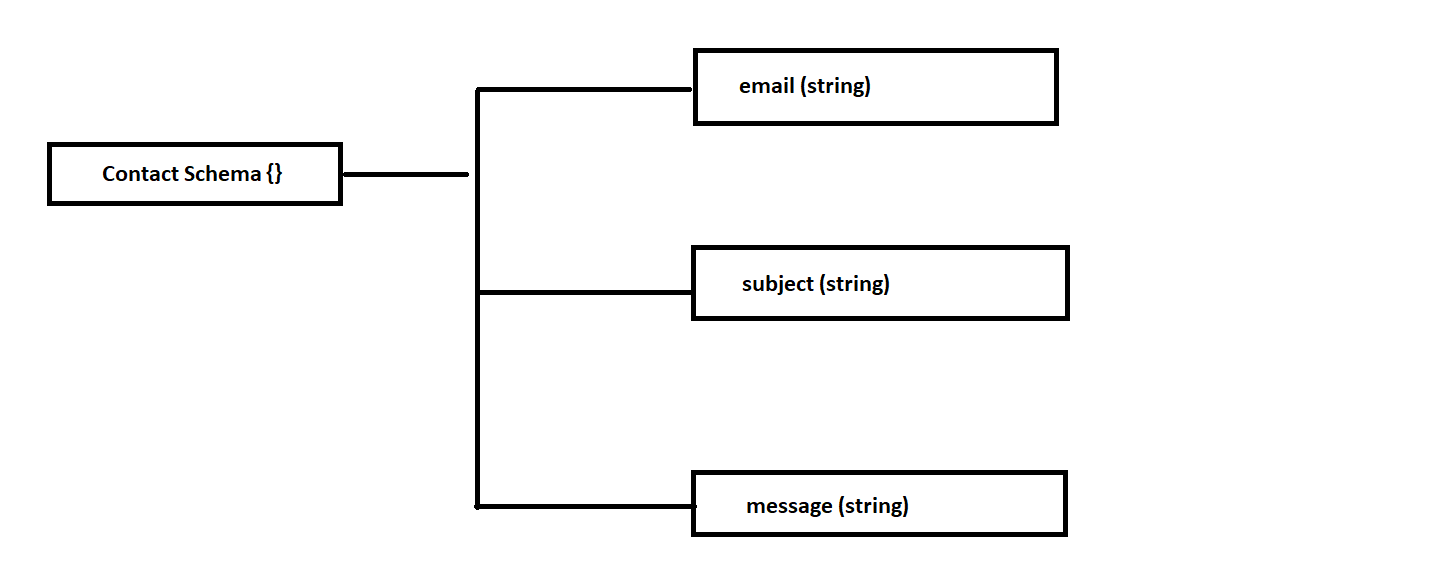
**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

****

**1. Create a new contact**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**2. Get all contact**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**3. Get individual contact by id**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**4. Reply to a contact message by provided email**

**AP**I:

**Type:**

**Method :** N/A

**Access**: **:** N/A

**Body :** N/A

**Params:** N/A

**Query:** N/A

**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**