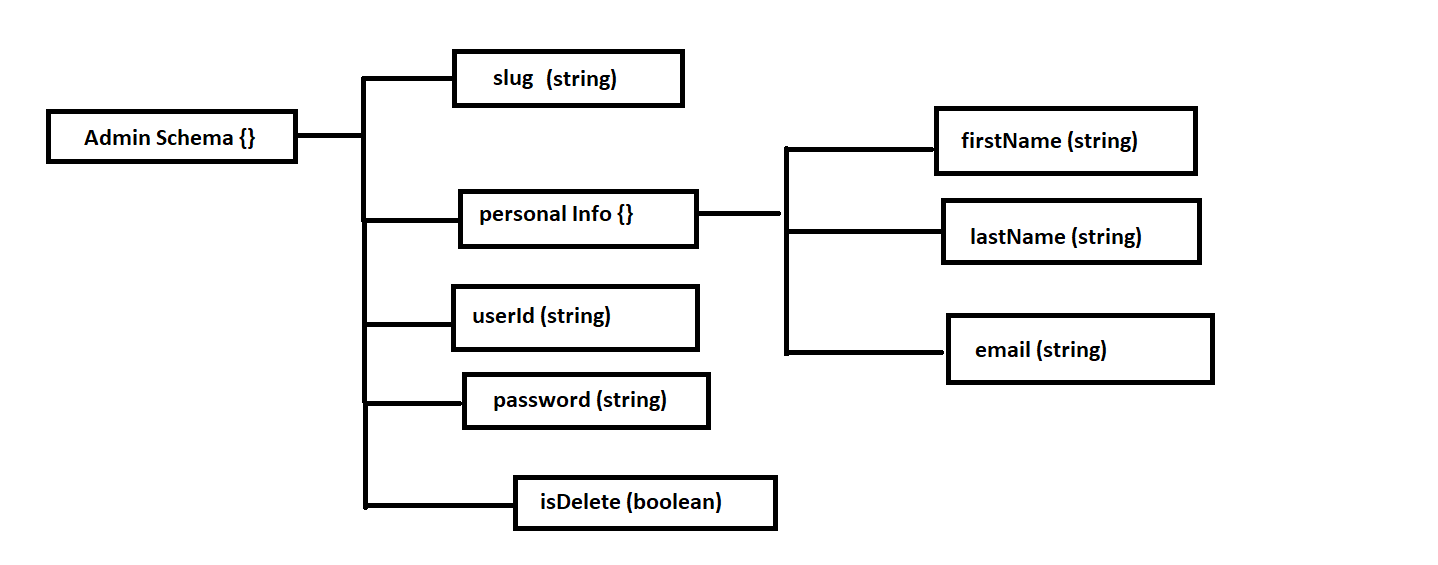
**API Documentation**

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**

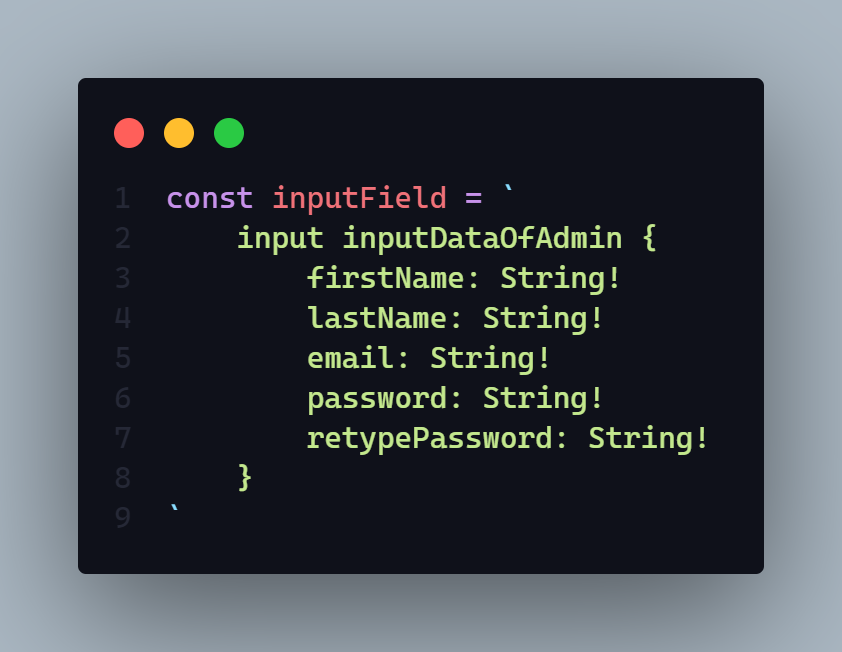
**Method :** POST / Mutation

**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**

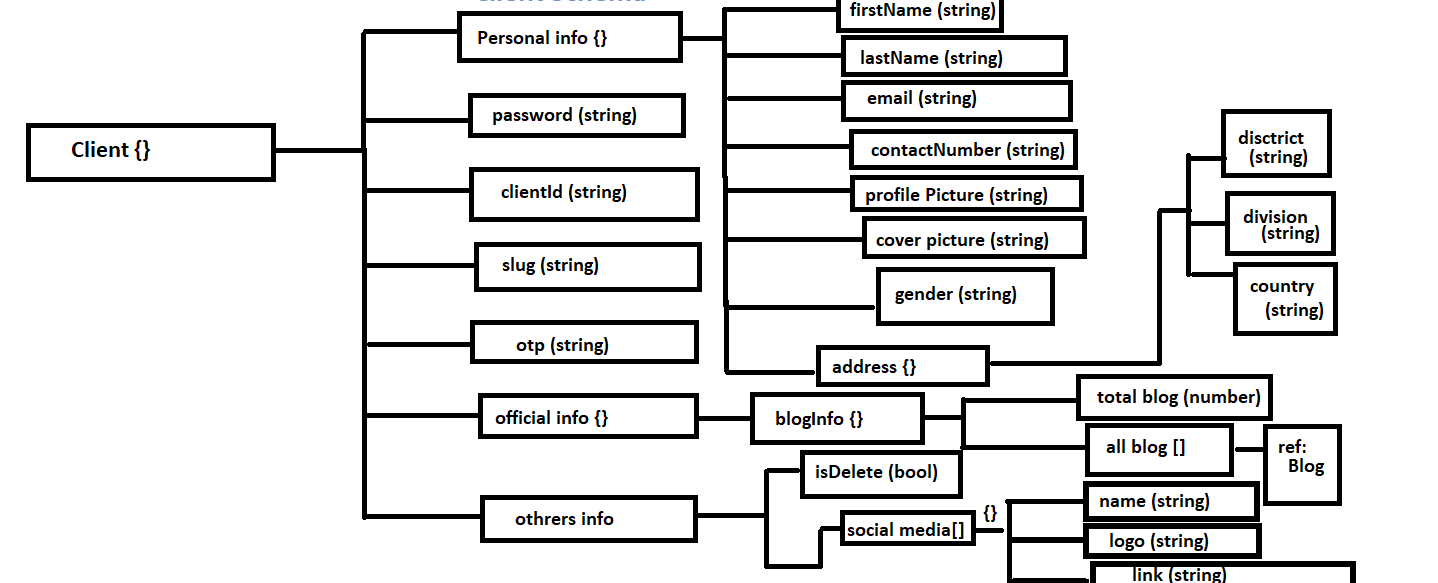
**Demo:**

****

**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**

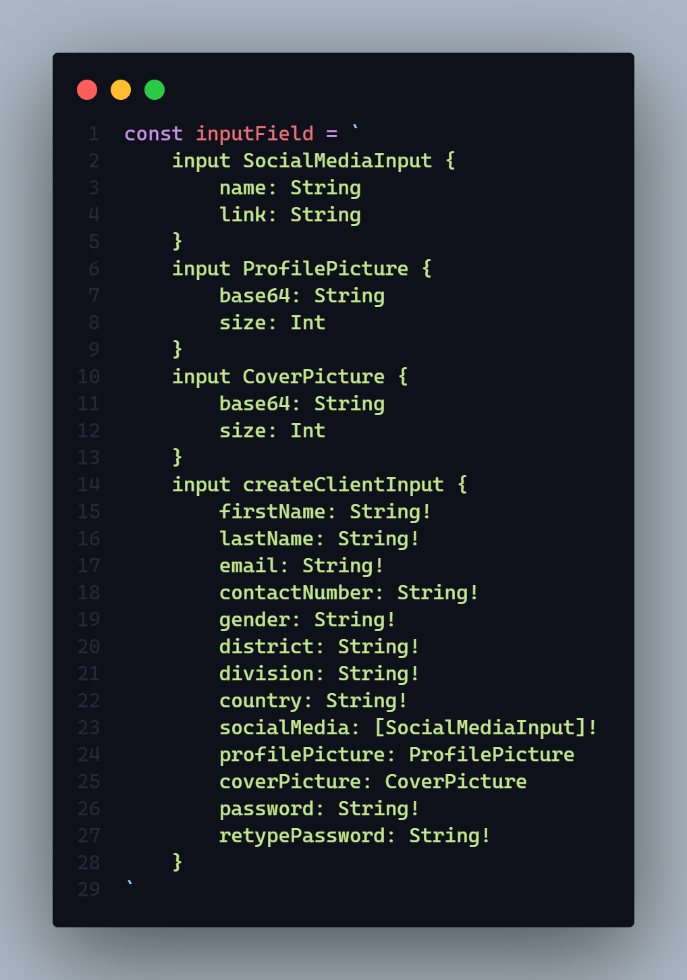
**AP**I: baseUrl/client eg: http://localhost:3030/client

**Type: GraphQl**

**Method :** POST / MUTATION

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

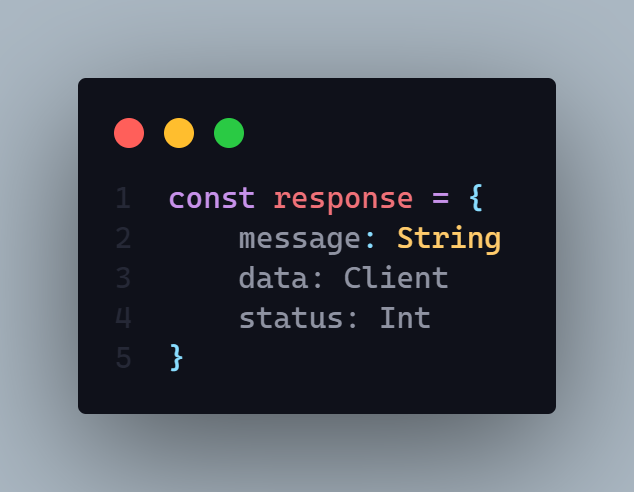
**Body :**



**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

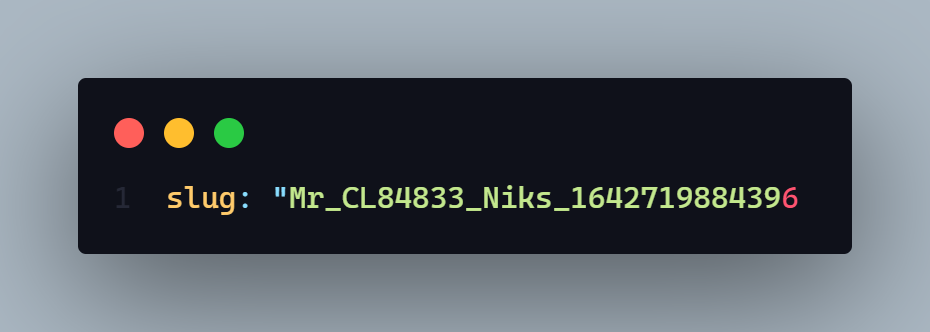
**Type: GraphQl**

**Method :**POST / MUTATION

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

**Auth : true**

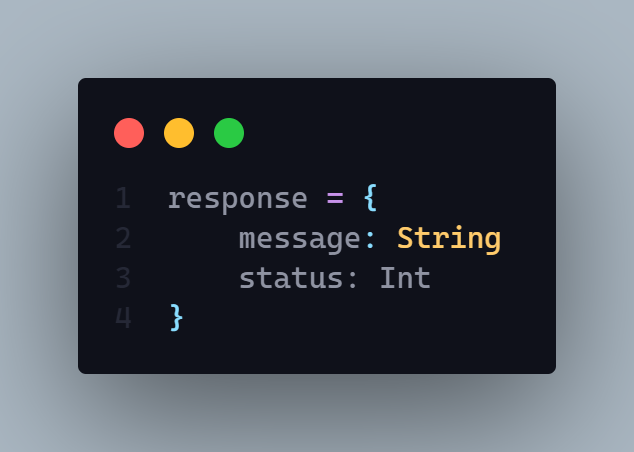
**Body :**



**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:**

****

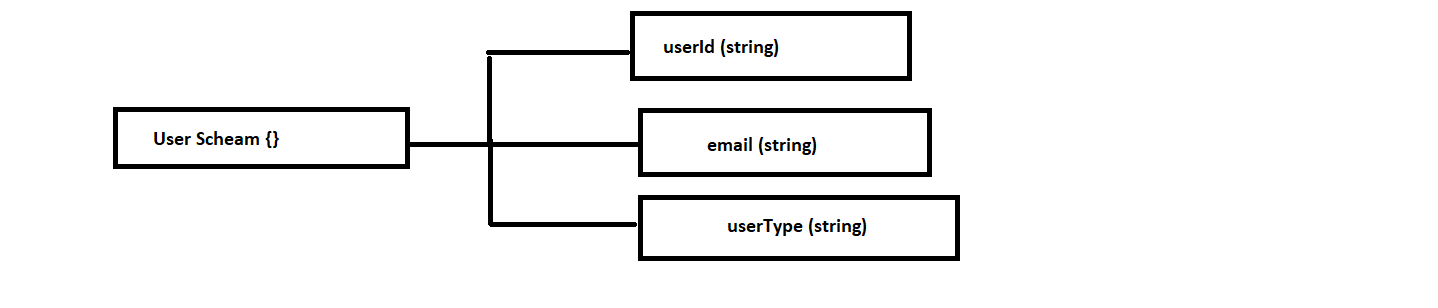
**Description:** If status code is

* **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)



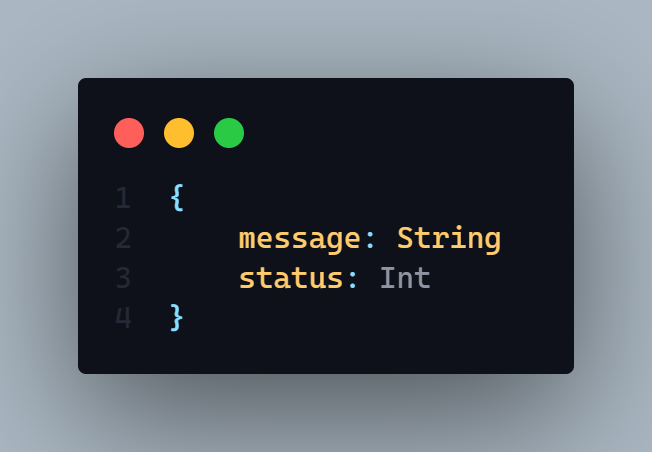
**Params:** N/A

**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

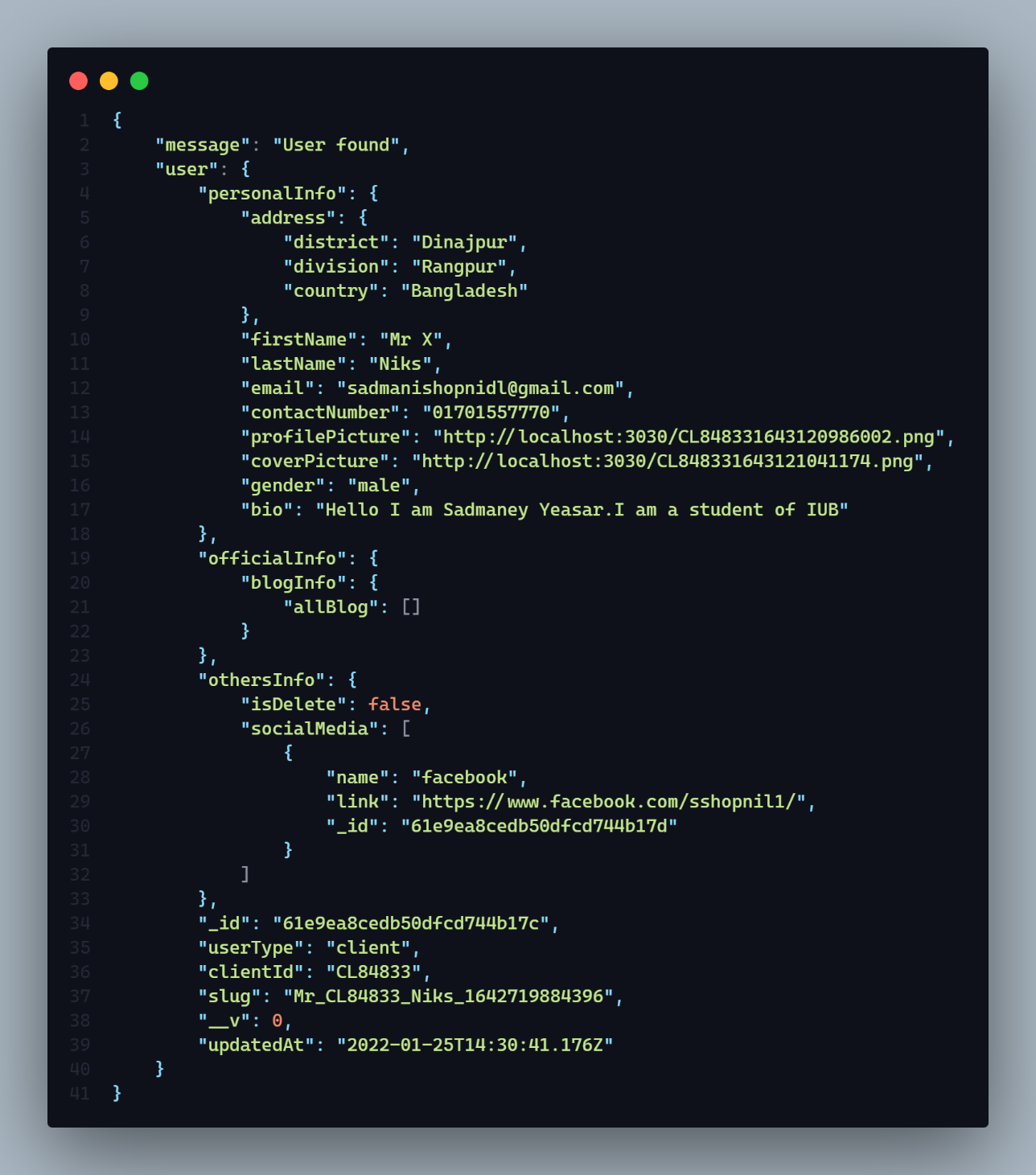
**Params:** N/A

**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**

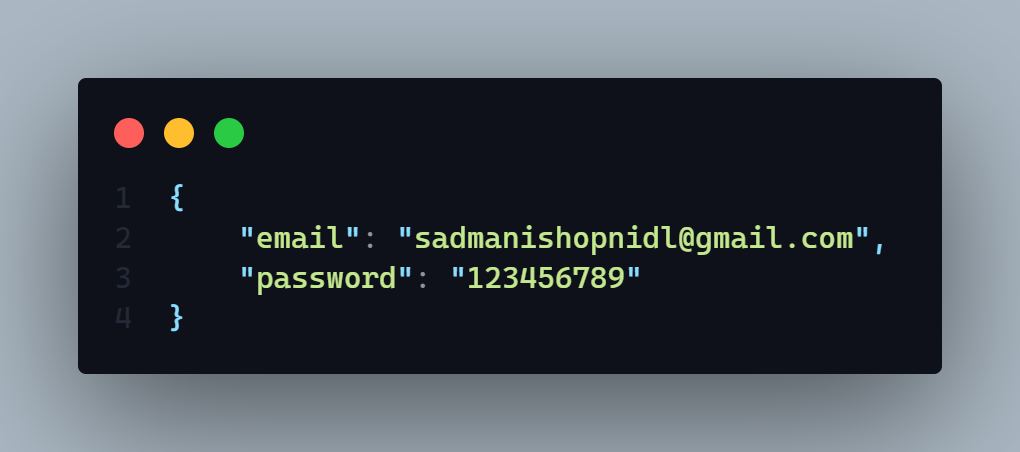
**AP**I: localhost:3030/user/login

**Type: REST**

**Method :** POST

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

**Body :**



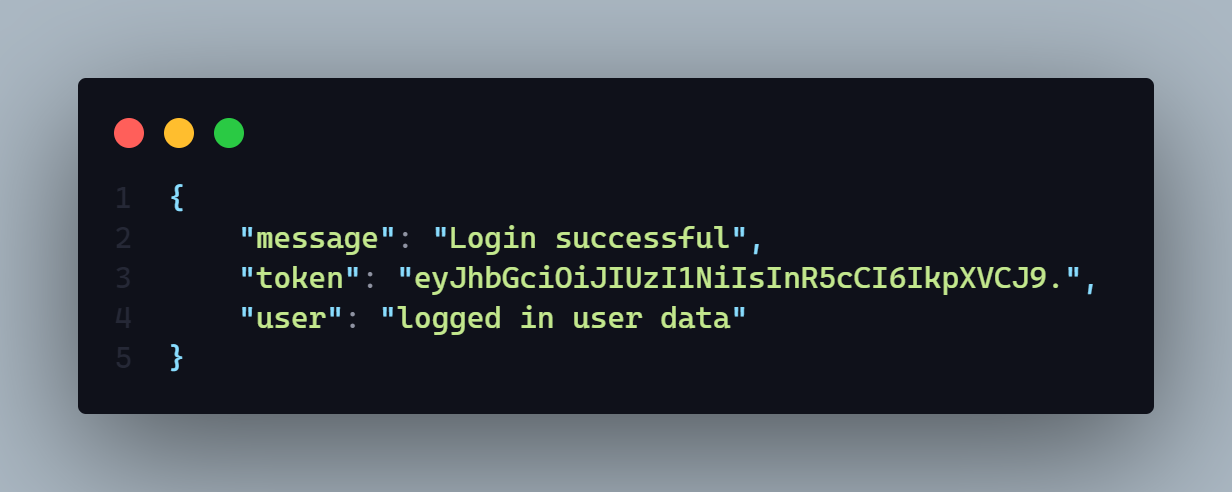
**Params:** N/A

**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

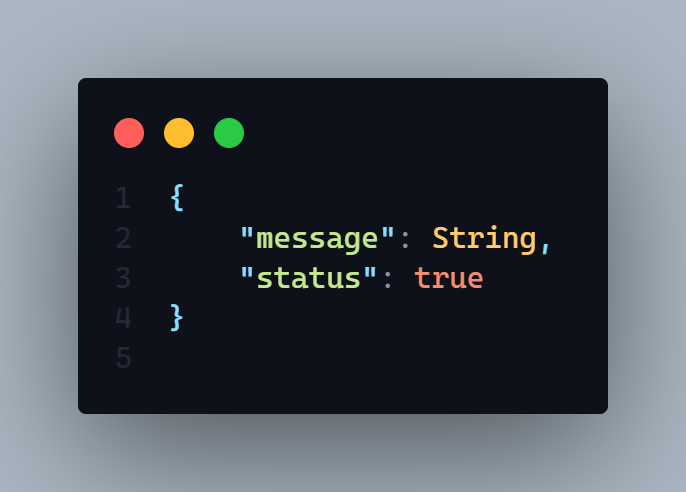


**Params:** N/A

**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 :**

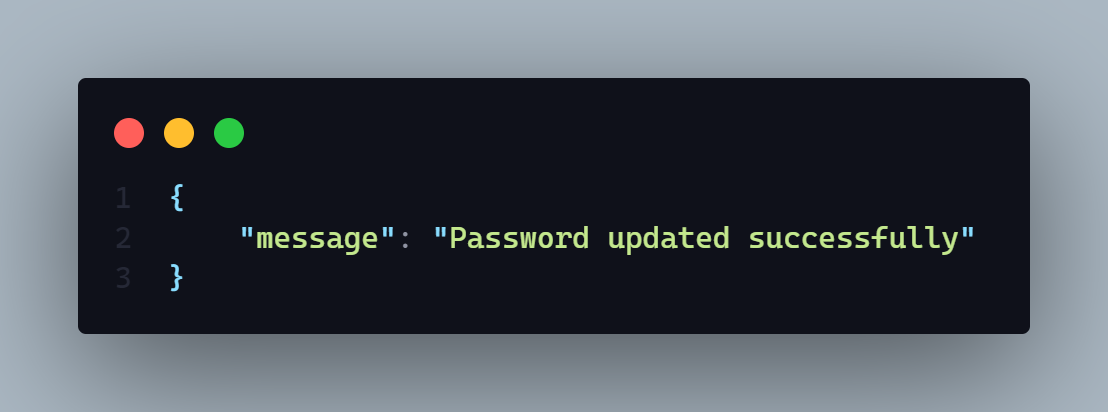


**Params:** N/A

**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

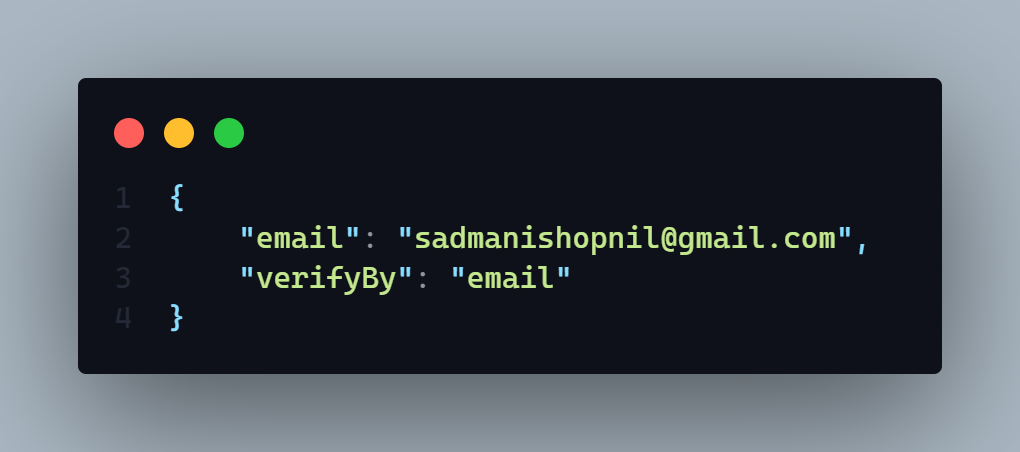
**Type: REST**

**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



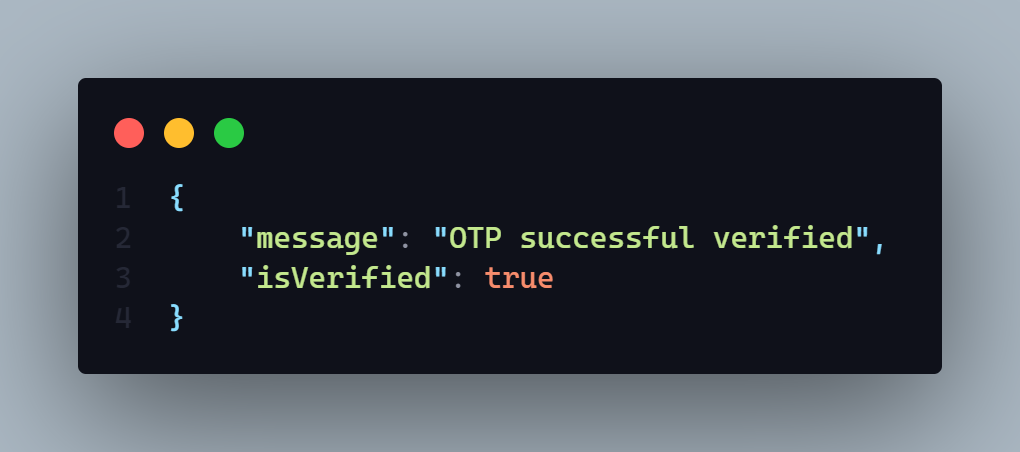
**Params:** N/A

**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

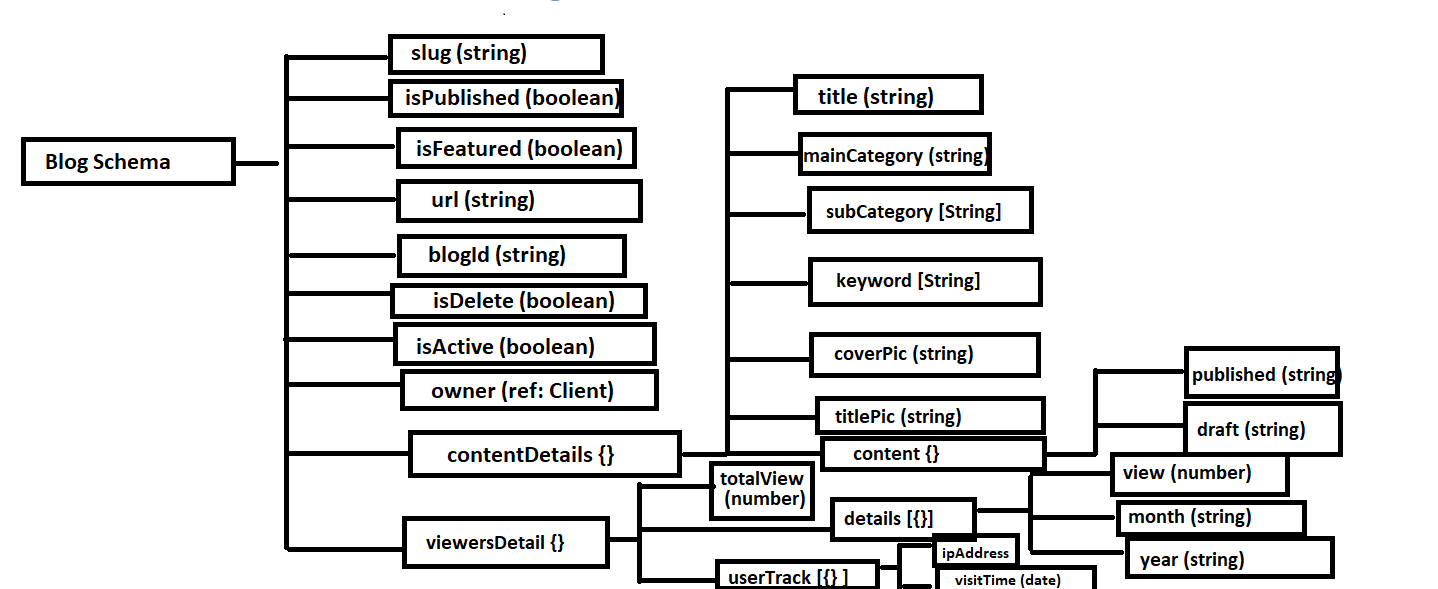
**Hints:** It will take the new password with retype password and change it

**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:

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

**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. Mark Blog as featured**

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

**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 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:

**Type:**

**Method :** N/A

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

**Body :** N/A

**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.

**Response:**

**Description:**

**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:

**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. 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

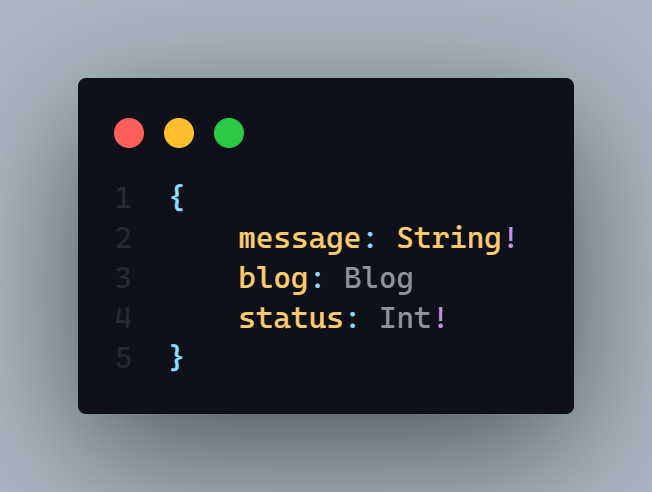
**Query:** N/A

**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 id** 
  + **It will query blog content from draft section**

**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:**

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

**12. Update blog element by id**

**1. sub category**

**2. Main Category**

**3. Keyword**

**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:**

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

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

**AP**I:

**Type:**

**Method :** N/A

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

**Body :** N/A

**Params:** N/A

**Query:** N/A

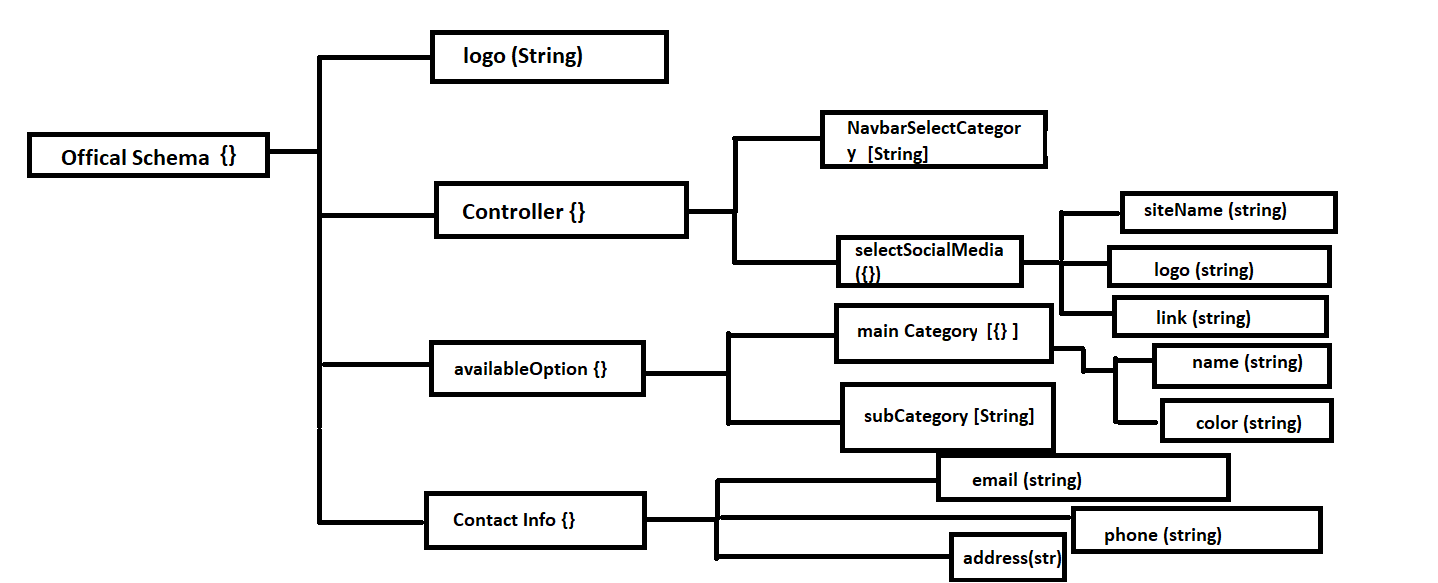
**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**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

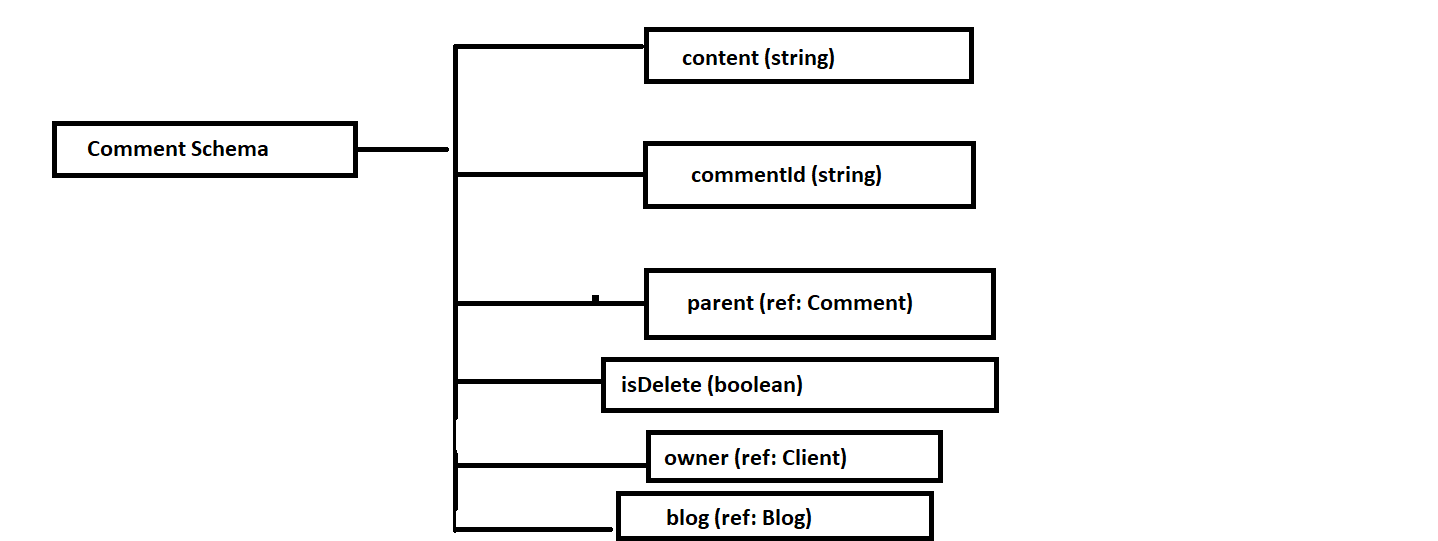
**Data:** N/A

**Hints:** N/A

**Response:**

**Description:**

**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

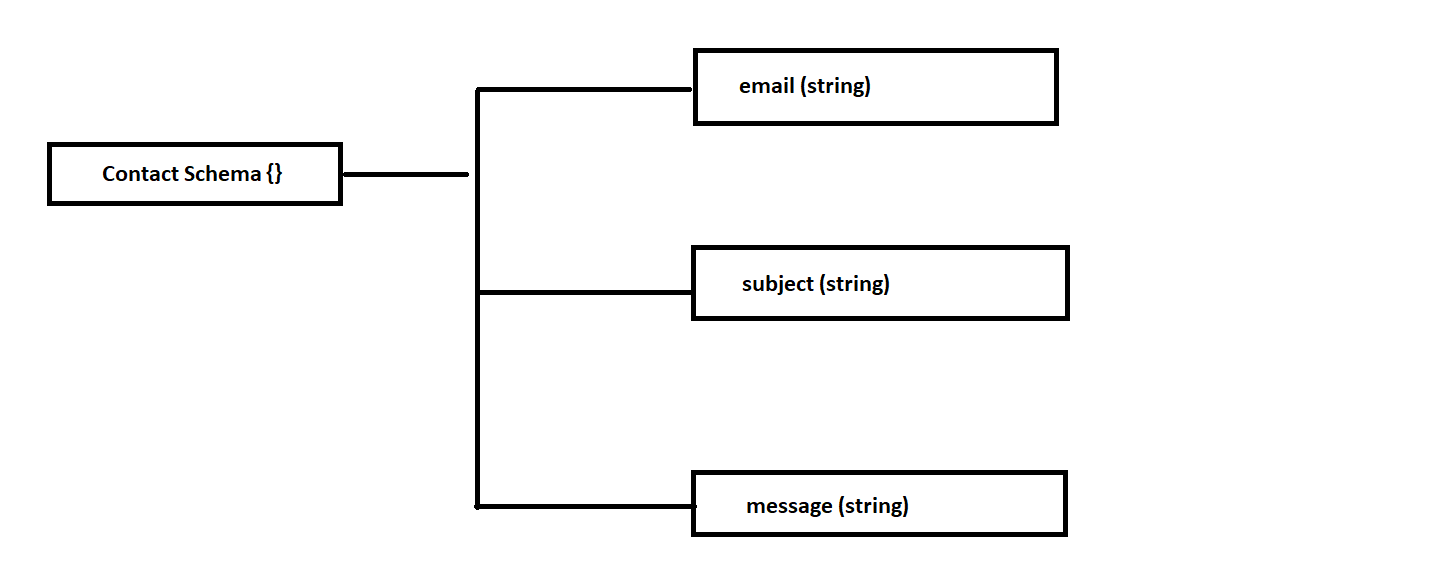
**Query:** N/A

**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:**