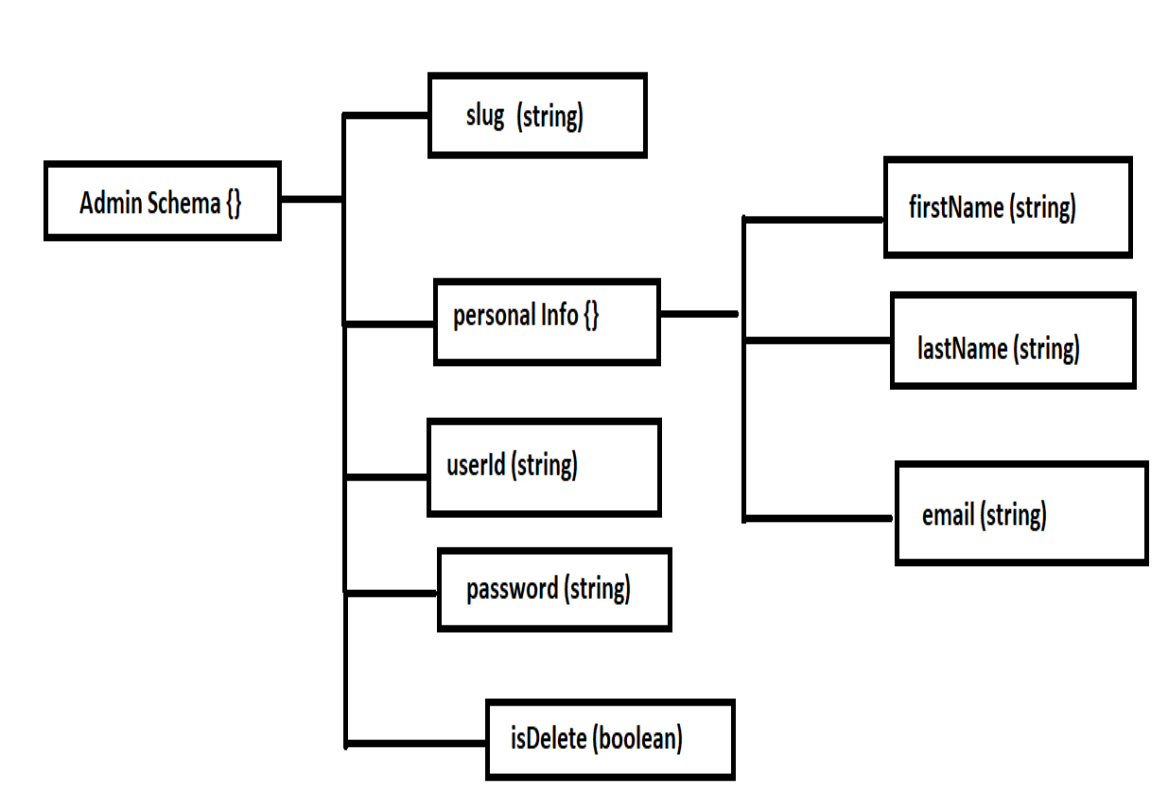
API Documentation

Technology Stack

Objective	Technology
Runtime	Node js
Server-side	Express js
Data Base	MongoDB (with ODM mongoose)
API Type	REST Full and GraphQL
Authentication	JWT
Email Sender	Node mailer
OTP Sender	TWILIO

TABLE OF CONTENTS

Admin Schema



1. Create a new Admin

API: baseUrl/admin eg: http://localhost:3030/admin

Type: GraphQl

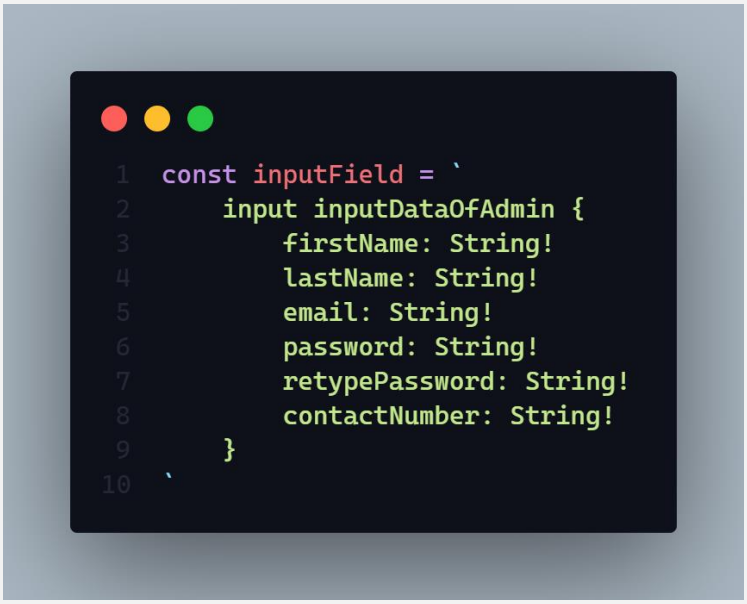
Method : POST / Mutation

Access: **:** All

Body :

Another Data =>

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



```
const inputField = `
    input inputDataOfAdmin {
        firstName: String!
        lastName: String!
        email: String!
        password: String!
        retypePassword: String!
        contactNumber: String!
    }
`
```

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**

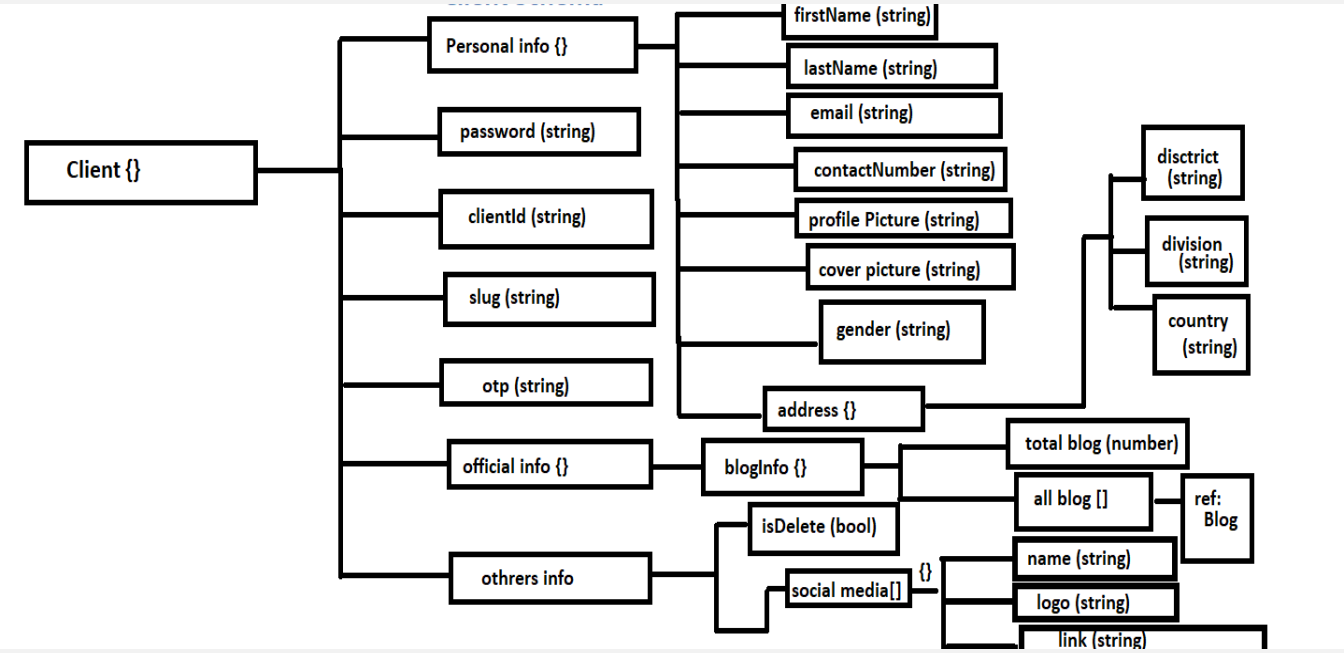
Demo:

```
mutation {
    createAdmin (input:{
        firstName:""
        lastName:""
        email:""
        password:""
        retypePassword:""
    }) {
    message,
    data{
        personalInfo{
            firstName
            lastName
        }
    },
    status
  }
}
```

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

API: baseUrl/client eg: http://localhost:3030/client

Type: GraphQl

Method : POST / MUTATION

Access: **:** client

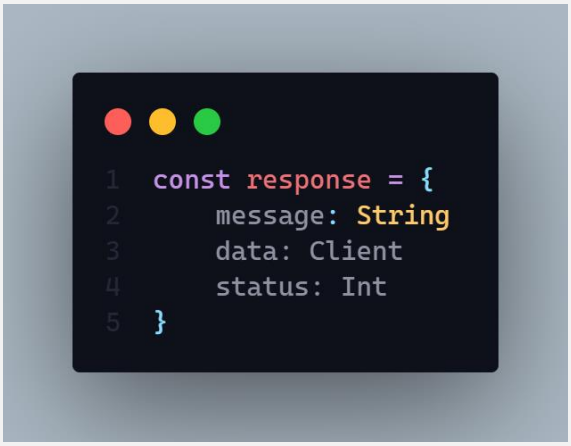
Body :

```
const inputField = `
    input SocialMediaInput {
        name: String
        link: String
    }
    input ProfilePicture {
        base64: String
        size: Int
    }
    input CoverPicture {
        base64: String
        size: Int
    }
    input createClientInput {
        firstName: String!
        lastName: String!
        email: String!
        contactNumber: String!
        gender: String!
        district: String!
        division: String!
        country: String!
        socialMedia: [SocialMediaInput]!
        profilePicture: ProfilePicture
        coverPicture: CoverPicture
        password: String!
        retypePassword: String!
    }
`
```

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:

```
mutation {
  createClient (input:{
    firstName:"Mr"
    lastName:"Niks"
    email:"sadmanishopnidl@gmail.com"
    gender:"male"
    district:"Dinajpur"
    division:"Rangpur"
    country:"Bangladesh"
    profilePicture:{
      base64: "",
      size: 0
    }

    coverPicture:{
      base64:"",
      size: 0
    }
    password:"123456789"
    retypePassword:"123456789"
    contactNumber:"01521427881"
    socialMedia: [
      {
        name:"facebook"
        link:"https://www.facebook.com/sshopnil1/"
      }
    ]
  }) {
    message,
    data {
      id
      personalInfo {
        firstName
        lastName
        email
        contactNumber
        profilePicture
        coverPicture
        gender
      }
    },
    status
  }
}
```

2. Delete a client by slug

API: baseUrl/client eg: http://localhost:3030/client

Type: GraphQl

Method : POST / MUTATION

Access: **:** admin, client

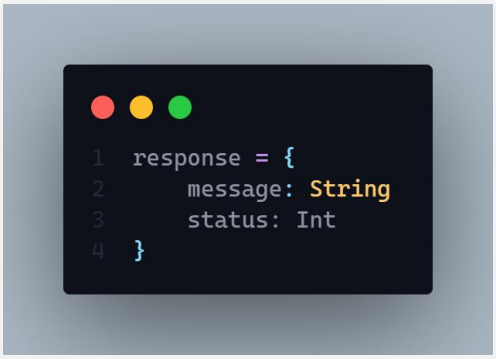
Auth : true

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.

API: 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:

```
response = {
    message: "No blogs has been found",
    data: [
        {
            month: "jan",
            no: 1,
            blogCount: 0
        }
        //till december
    ]
}
```

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

API: localhost:3030/client

Type: GraphQL

Method : POST / MUTATION

Access: **:** admin, client

Auth : true

Body :

```
updateClient(slug: "Mr_CL84833_Niks_1642719884396", data: {
    firstName: String,
    lastName: String,
    contactNumber: String,
    district : String,
    division: String,
    country: String,
    socialMedia: [
        {
            name: String,
            link: String
        }
    ]
)
```

Params: N/A

Query: N/A

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

Hints: N/A

Response:

```
{
    "data": {
        "updateClient": {
            "message": "Client updated successfully",
            "status": 202
        }
    }
}
```

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

```
mutation {
  updateClient(slug: "Mr_CL84833_Niks_1642719884396", data: {
    bio: "Hello I am Sadmaney Yeasar.I am a student of IUB"
    firstName: String
    lastName : String
    contactNumber : String
    district : String
    division : String
    country : String
    socialMedia [
        {
            name: String
            link: String
        }
    ]

  }) {
    message
    status
  }
}
```

5. Client can see only his published or draft blog

API: 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”)

```
mutation {
    seeOwnBlog (input: {
        sortBy: "",
        pageNo: 1
        limit: 4
        searchFor: "draft"
    }
)
```

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

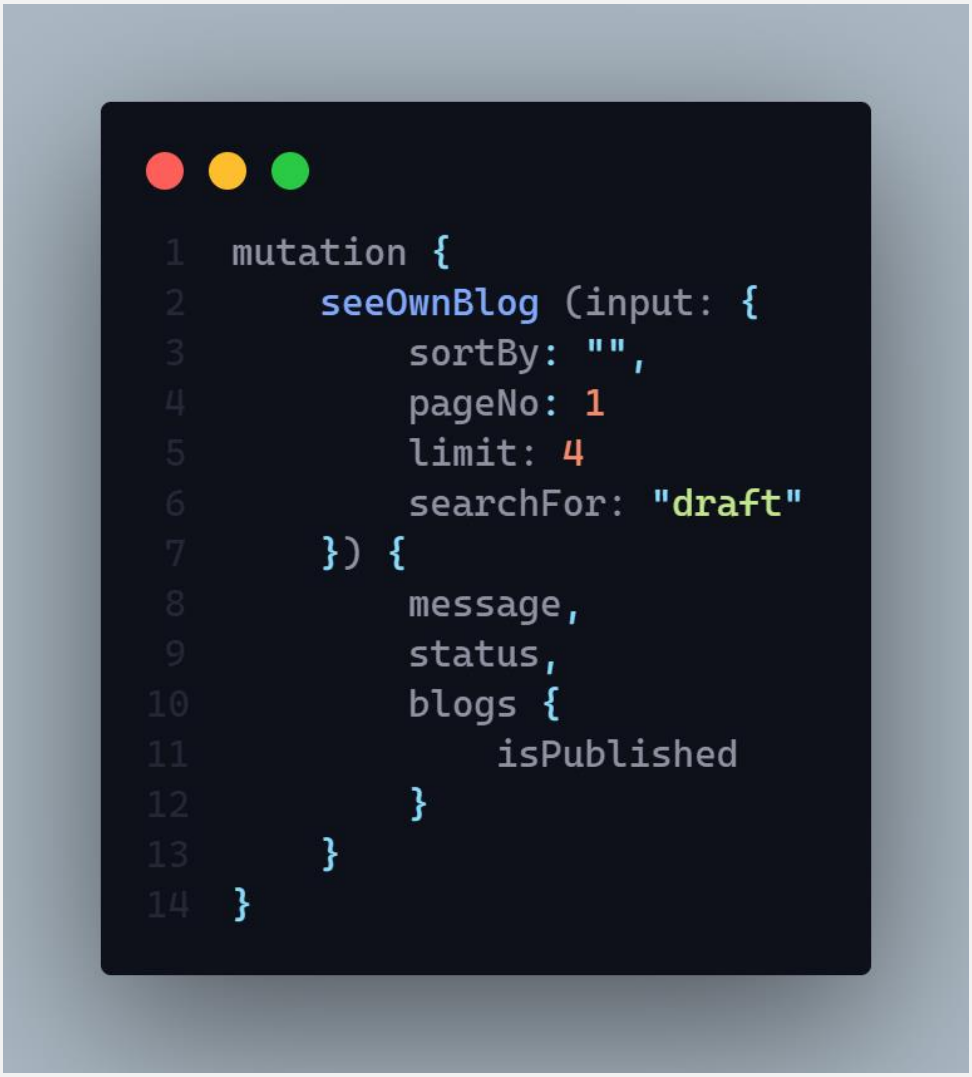
Response:

```
{
    "data": {
        "seeOwnBlog": {
            "message": "0 blog found",
            "status": 202,
            "blogs": []
        }
    }
}
```

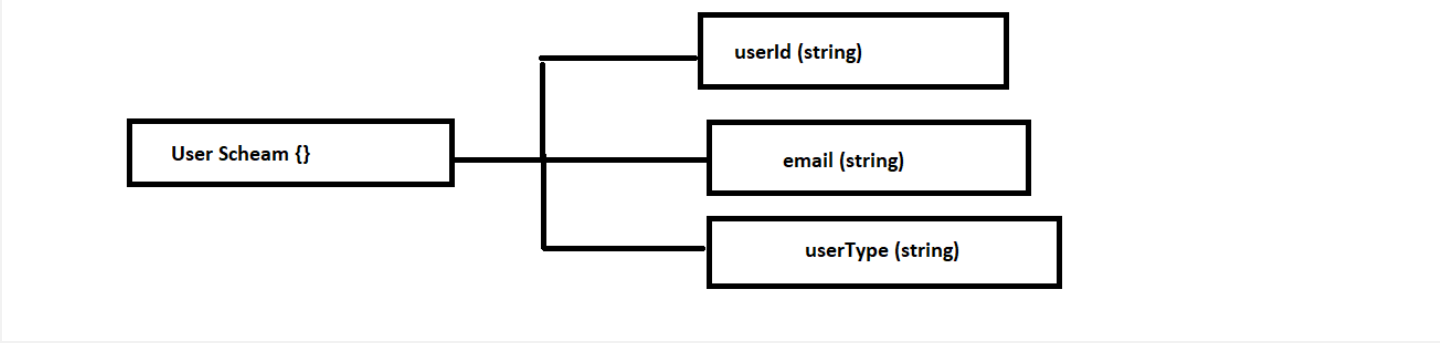
Description: If status code is

- **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 :


mutation {
seeOwnBlog (input: {
sortBy: "",
pageNo: 1
limit: 4
searchFor: "draft"
}) {
message,
status,
blogs {
isPublished
}
}
}

User Schema



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

API: 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)

```
mutation {
    updateProfileImage (input: {
        uploadType : Stirng //value = cover or profile
        base64: String,
        size: Int
    })
}
```

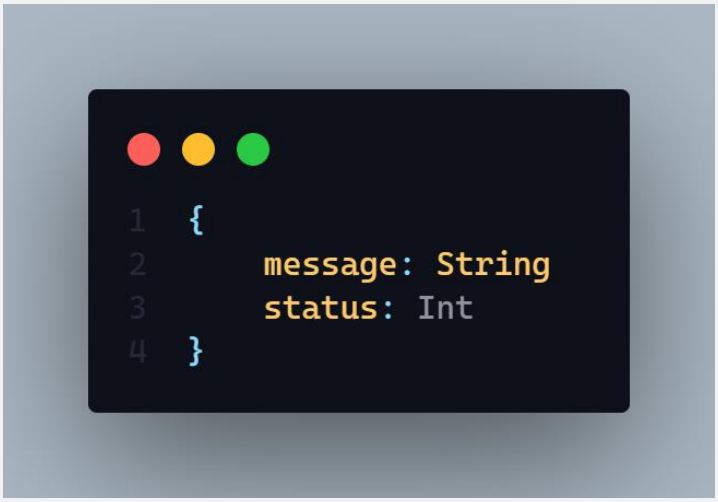
Params: N/A

Query: N/A

Hints: N/A

Response:

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



```
{
    message: String
    status: Int
}
```

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:

```
mutation {
    updateProfileImage (input: {
        uploadType : "cover"
        base64: "",
        size: 54581
    }) {
        message
        status
    }
}
```

2. User can see his profile

API: localhost:3030/user/profile

Type: GraphQL

Method : GET

Access: **:** All

Auth : true

Body : N/A

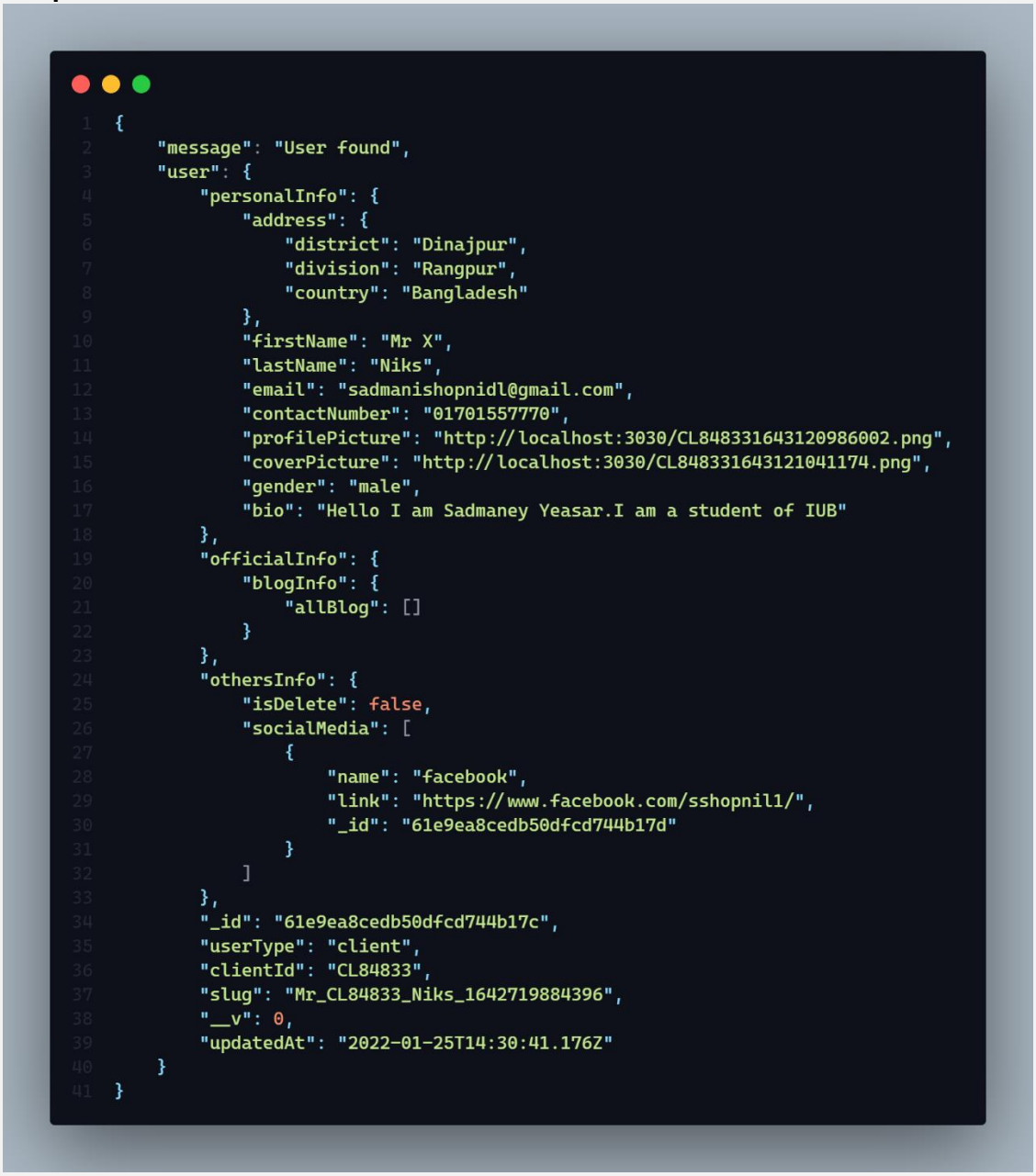
Params: N/A

Query: N/A

Data: N/A

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

Response:




```
{
    "message": "User found",
    "user": {
        "personalInfo": {
            "address": {
                "district": "Dinajpur",
                "division": "Rangpur",
                "country": "Bangladesh"
            },
            "firstName": "Mr X",
            "lastName": "Niks",
            "email": "sadmanishopnidl@gmail.com",
            "contactNumber": "01701557770",
            "profilePicture": "http://localhost:3030/CL848331643120986002.png",
            "coverPicture": "http://localhost:3030/CL848331643121041174.png",
            "gender": "male",
            "bio": "Hello I am Sadmaney Yeasar.I am a student of IUB"
        },
        "officialInfo": {
            "blogInfo": {
                "allBlog": []
            }
        },
        "othersInfo": {
            "isDelete": false,
            "socialMedia": [
                {
                    "name": "facebook",
                    "link": "https://www.facebook.com/sshopnil1/",
                    "_id": "61e9ea8cedb50dfcd744b17d"
                }
            ]
        },
        "_id": "61e9ea8cedb50dfcd744b17c",
        "userType": "client",
        "clientId": "CL84833",
        "slug": "Mr_CL84833_Niks_1642719884396",
        "__v": 0,
        "updatedAt": "2022-01-25T14:30:41.176Z"
    }
}
```


Description: : If status code is

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

3. Login user with email and password

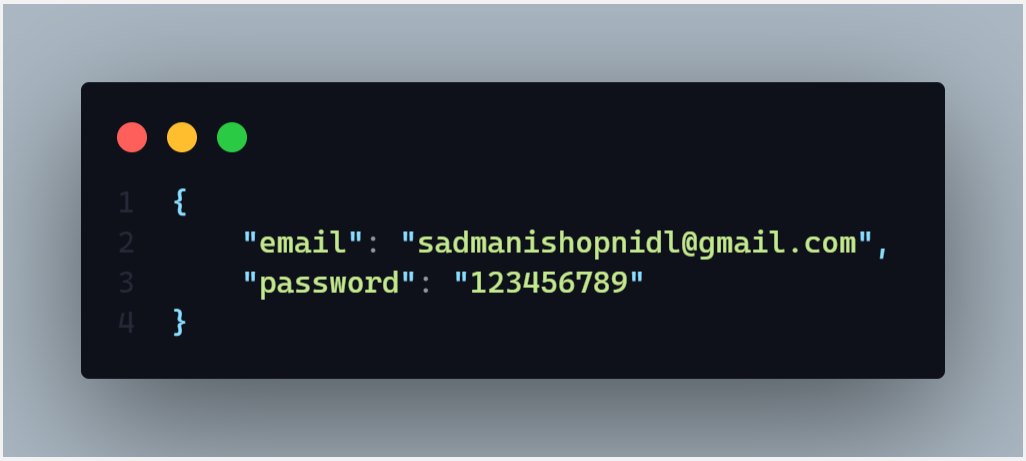
API: localhost:3030/user/login

Type: REST

Method : POST

Access: **:** ALL

Body :



```
{
    "email": "sadmanishopnidl@gmail.com",
    "password": "123456789"
}
```

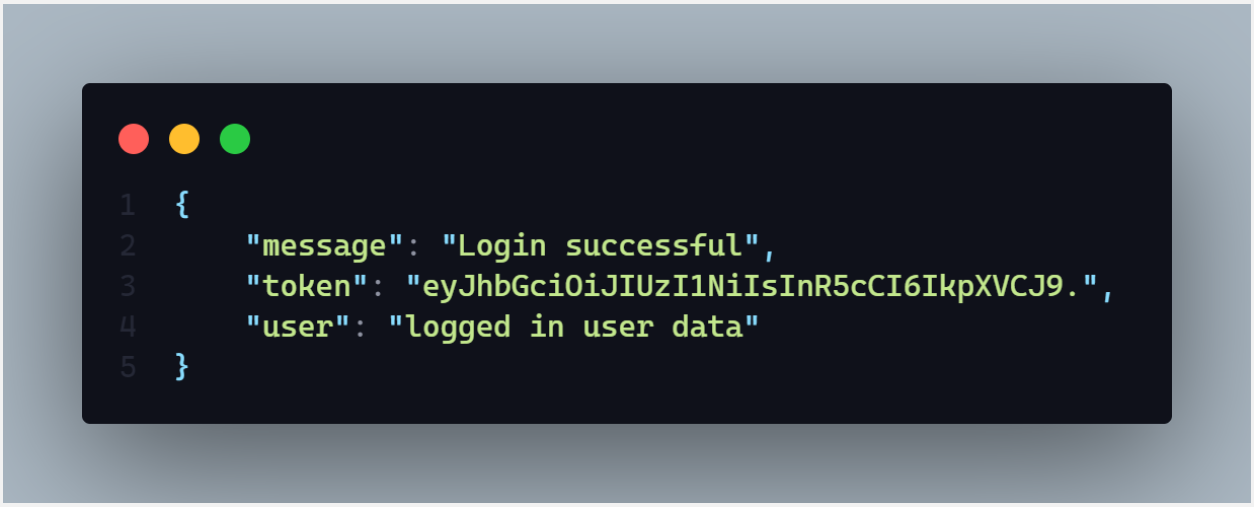
Params: N/A

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:



```
{
    "message": "Login successful",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.",
    "user": "logged in user data"
}
```

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

API: localhost:3030/client/verify

Type: REST

Method : POST

Access: **:** admin, client

Auth : true

Body :

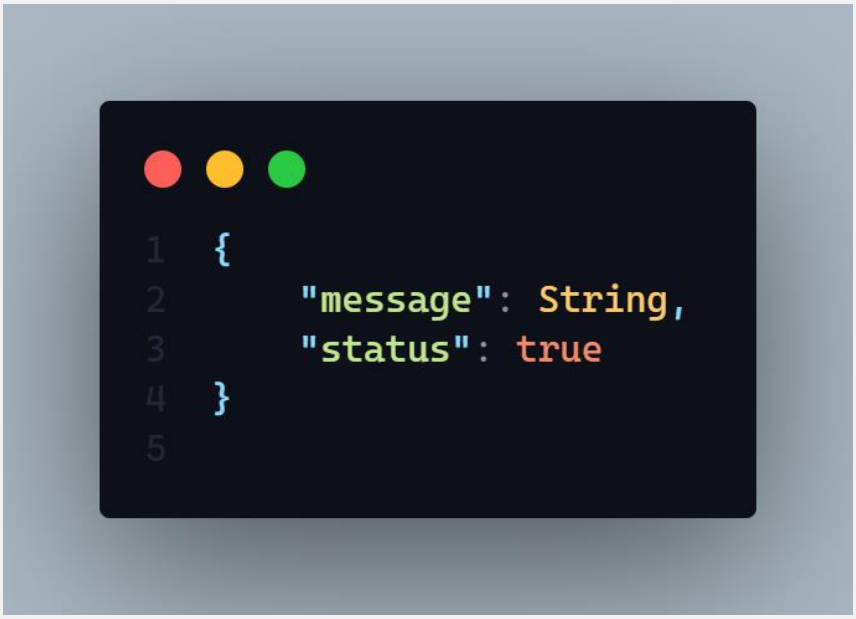
1. Password => String

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)

API: localhost:3030/user/update/password

Type: REST

Method : PUT

Access: **:** admin, client

Auth: true

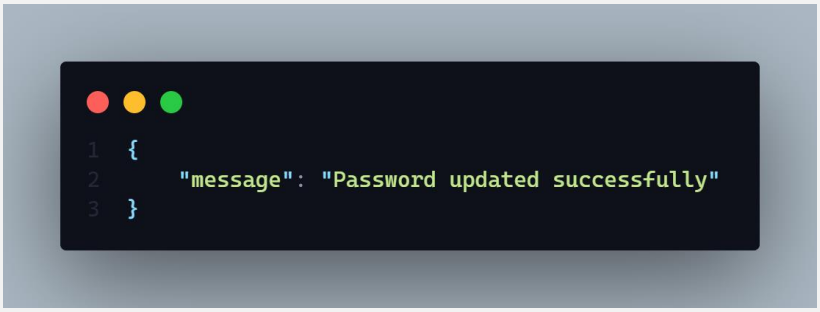
Body :

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)

API: localhost:3030/user/forgot/password

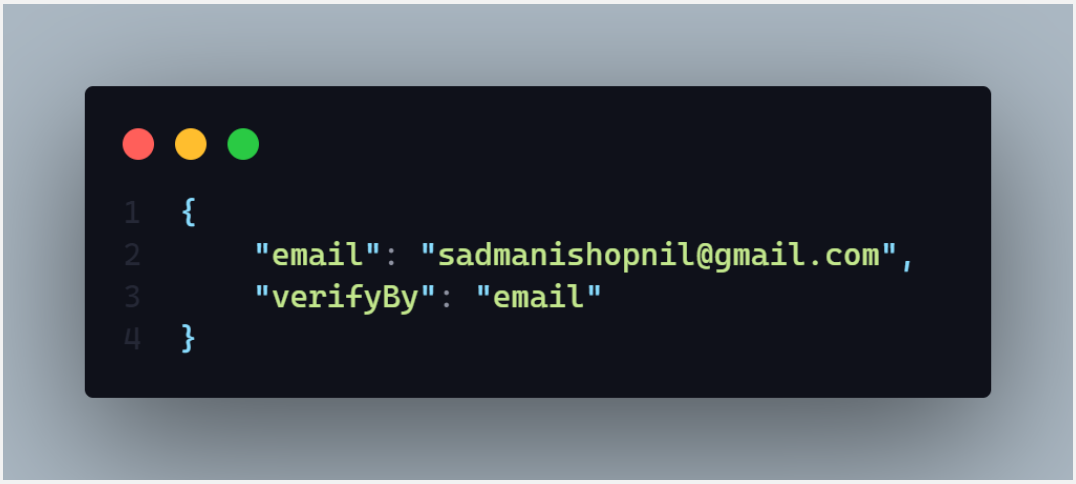
Type: REST

Method : POST

Access: **:** N/A

Auth: false

Body :



```
{
    "email": "sadmanishopnil@gmail.com",
    "verifyBy": "email"
}
```

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:

```
{
    "message": "A 4 digit otp has been sent to sadmanishopnil@gmail.com"
}
```

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.

API: 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:

```
{
    "message": "OTP successful verified",
    "isVerified": true
}
```

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.**

API: localhost:3030/user/reset/password

Type: REST

Method : POST

Access: **:** ALL

Body :

Params: N/A

Auth: false

Query: N/A

Data: N/A

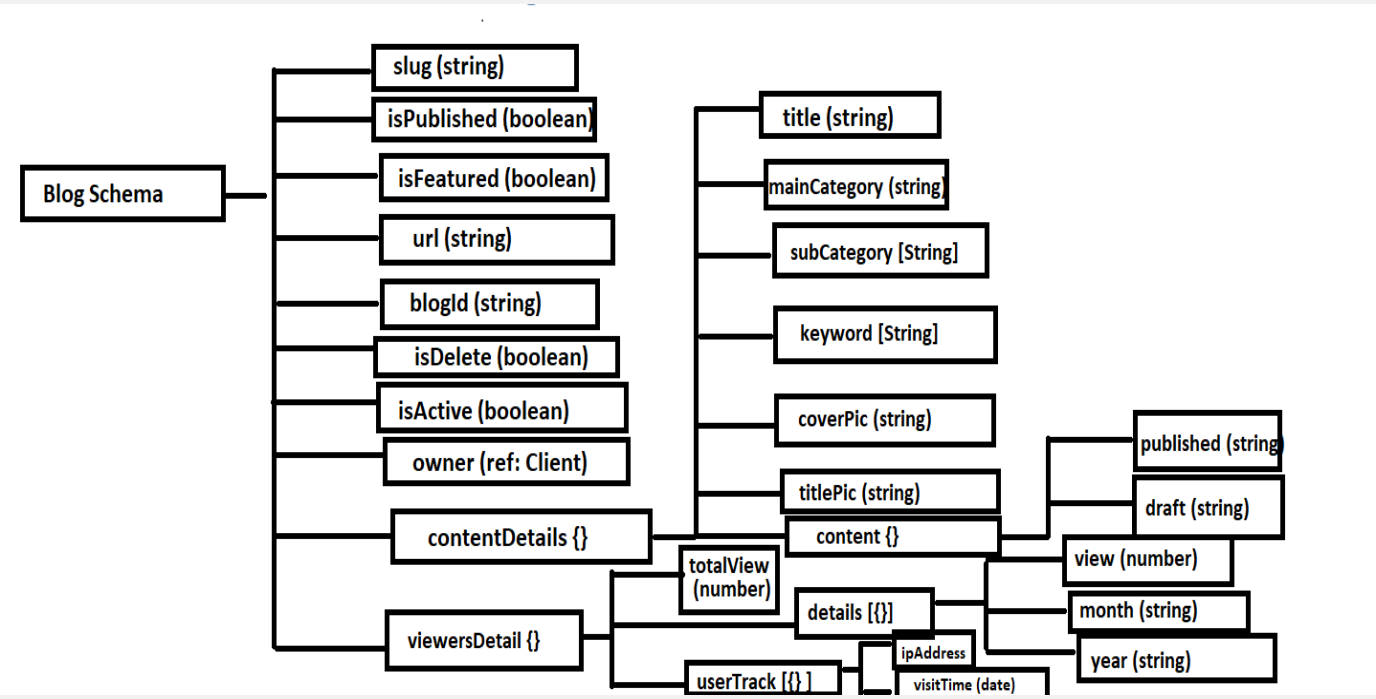
Hints: It will take the new password with retype password and change it

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**

API: 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 =>**
   a. queryBy : String [expected Data => **"recent"**]
2. If User want to get **Top of this month =>**
   a. queryBy : String [expected Data => **"topMonth"**]
3. If User want to get **Most Read blog =>**
   a. queryBy : String [expected Data => **"mostRead"**]
4. If User want to get all **Featured Blog =>**
   a. queryBy : String [expected Data => **"featured"**]
5. If User want to get blog by **Main Category =>**
   a. queryBy : String [expected Data => **"mainCategory"**]
   b. 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 =>**
   a. filter By : {
   publishedYear: String [expected data => "2022" or "2021" …..]
   }
2. If User want to filter blog by **Sub Category =>**
   a. filter By : {
   subCategory : [String] [expected data => ["web Development", "web design"]]
   }

Search Part

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

Sorting Part

1. If user want to sort blog =>
    a. sortBy : String [Expected input will be a string]
        i. option => 1
            1. latest
            2. A_Z
            3. Z_A
            4. View

Page No:

1. **If** user want to give page number = >
    a. **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**
    a. **dataLimit: String [** Expected input will be a string of integer number like "5" **] it means server will response back 5 data each page.**
        i. **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:

```
blogs (
        queryBy: "", //expected data recent , topMonth ,mostRead,featured,mainCategory,mostRead
        queryInput: "Web Design"
        search : ""
        filterBy: {
            publishedYear: "2022",
            subCategory: [
                "web"
            ]
        }
        sortBy: "latest" //expected data will be ⇒ latest, A_Z, Z_A, view
        pageNo: 2 //this will indicate the page no
        dataLimit: "1" //this will set the data limit that how many data it will return in each time
    )
```

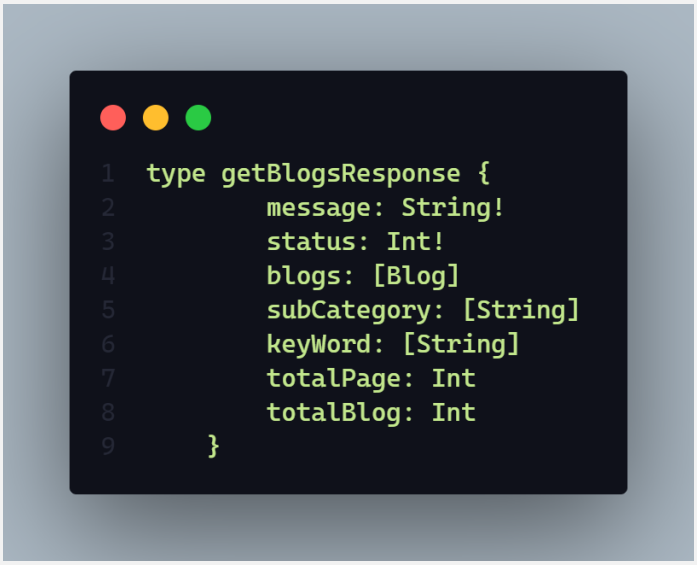
Params: N/A

Query: N/A

Data: N/A

Hints: N/A

Response:



```
type getBlogsResponse {
        message: String!
        status: Int!
        blogs: [Blog]
        subCategory: [String]
        keyWord: [String]
        totalPage: Int
        totalBlog: Int
    }
```

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:

```
query {
    blogs (
        queryBy: "", //not mandatory
        queryInput: "Web Design"  //not mandatory
        search : "" //not mandatory
        filterBy: {  //not mandatory
            publishedYear: "2022",
            subCategory: [
                "web"
            ]
        }
        sortBy: "latest" //not mandatory
        pageNo: 2 //not mandatory
        dataLimit: "1" //not mandatory
    ){
        message //mandatory
        status //mandatory
        subCategory //not mandatory
        keyWord //not mandatory
        totalPage //not mandatory
        totalBlog //not mandatory
        blogs { //mandatory
            slug
            owner {
                personalInfo {
                    firstName
                    lastName
                    coverPicture
                }
            }
        }
    }
}
```

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

API: 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:

```
type getTopMonthBlogResponse {
        message: String!
        category : [String]
        blog : [Blog]
        status: Int!
    }
```

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
    a. 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

```
query {
    getTopMonthBlog {
        message
        status
        blog {
            contentDetails {
                mainCategory
                title
                subCategory
                keyWord
                coverPic
                titlePic
                content {
                    published
                    draft
                }
            }
            viewersDetails {
                totalView
                details {
                        view
                        month
                        year
                    }
            }
        }
        category
    }
}
```

3. Mark Blog as featured

API: localhost:3030/blog

Type: GraphQL

Method : POST / MUTATIOn

Access: **:** admin

Auth: true

Body : Need to pass =>

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

```
setIsFetcherBlog (slugs : [String]!) : IsFetcherResponse
```

Params: N/A

Query: N/A

Data: N/A

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

Response:

```
{
    "data": {
        "setIsFetcherBlog": {
            "message": "Blog has successfully featured",
            "status": 202
        }
    }
}
```

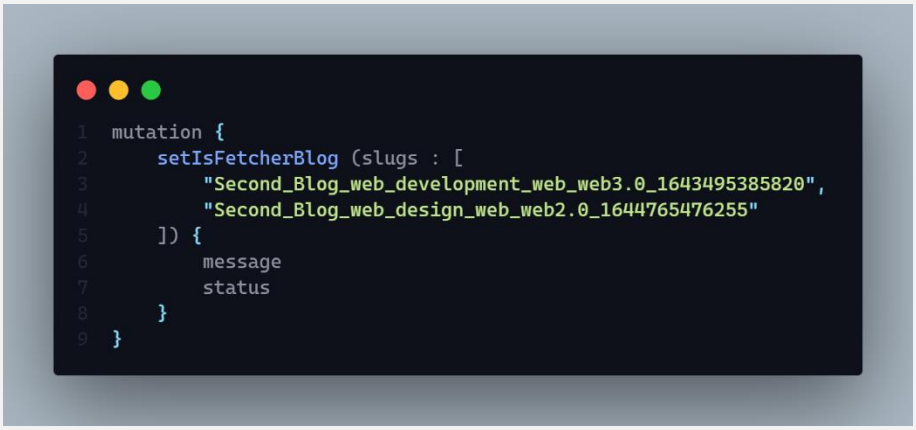
Description: : If status code is

- **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

API: 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

```
countViewers (slug: String) : countViewersResponse
```

Params: N/A

Query: N/A

Data: N/A

Hints: N/A

Response:

```
type countViewersResponse {
        message: String!
        status: Int!
    }
```

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 =>
    a. Logged in user can not increase own blog view
    b. View will be count of a user only one time in a day by tracking the user device ip address.
    c. 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 (incomplete)

API:

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 .

API: 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 *

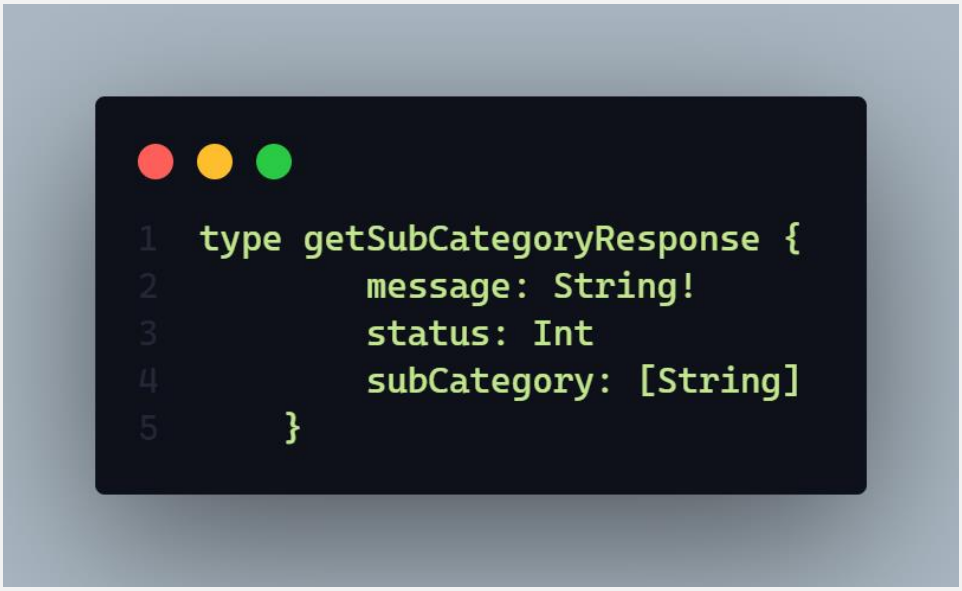
2. Output response:

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:



```
type getSubCategoryResponse {
        message: String!
        status: Int
        subCategory: [String]
    }
```

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 :

```
query {
    getSubCategory (category: "Development") {
        message,
        status,
        subCategory
    }
}
```

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

API: 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:

```
type getIndividualBlogResponse {
        message: String!
        status: Int!
        blog: Blog
        relatedBlog: [Blog]
    }
```

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
    a. Blog main category
    b. Blog keyword
    c. Blog sub category
5. All related blog query action in or process
6. It only show published and active blog

Demo :

```
query {
    getIndividualBlog (
        slug: "Second_Blog_web_design_web_web2.0_16453058761 40" //mandatory
    ){
        message //mandatory
        status  //mandatory
        blog {
            slug
            contentDetails {
                title
            }
            blogId
        }
        relatedBlog {
            slug
            contentDetails {
                title
            }
            blogId
        }
    }
}
```

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**

API: localhost:3030/blog

Type: GraphQL

Method : POST

Access: **:** client

Body :

```
saveBlog (
        input: {
            title: "First Blog", **
            mainCategory: "Web Development", **
            subCategory: ["web"],
            keyWord: ["web"],
            content: "jflsdjflskfjsfjsfjlkksldfjskldfjlskdfjlskdf" **
            coverPic: {
                base64: "",
                size: 2555
            },
            titlePic: {
                base64: ""
                size: 2588
            }
        }
        blogId: "BLG56648"
    )
```

Params: N/A

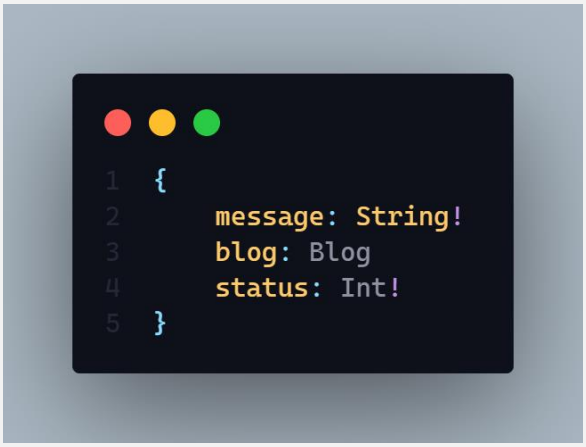
Query: N/A

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:

```
mutation {
    saveBlog (
        input: {
            title: "First Blog", **
            mainCategory: "Web Development", **
            subCategory: ["web"],
            keyWord: ["web"],
            content: "jflsdjflskfjsfjsfjlkksldfjskldfjlskdfjlskdf" **
            coverPic: {
                base64: "",
                size: 2555
            },
            titlePic: {
                base64: ""
                size: 2588
            }
        }
        blogId: "BLG56648"
    ) {
        message, **
        blog,
        status **
    }
}
```

(**) 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.**

API: localhost:3030/blog

Type: GraphQL

Method : POST

Access: **:** client

Auth: true

Body :

```
saveBlog (
        input: {
            title: "First Blog", **
            mainCategory: "Web Development", **
            subCategory: ["web"],
            keyWord: ["web"],
            content: "jflsdjflskfjsfjsfjlkksldfjskldfjlskdfjlskdf" **
            coverPic: {
                base64: "",
                size: 2555
            },
            titlePic: {
                base64: ""
                size: 2588
            }
        }
        blogId: "BLG56648"
    )
```

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**

API: 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:

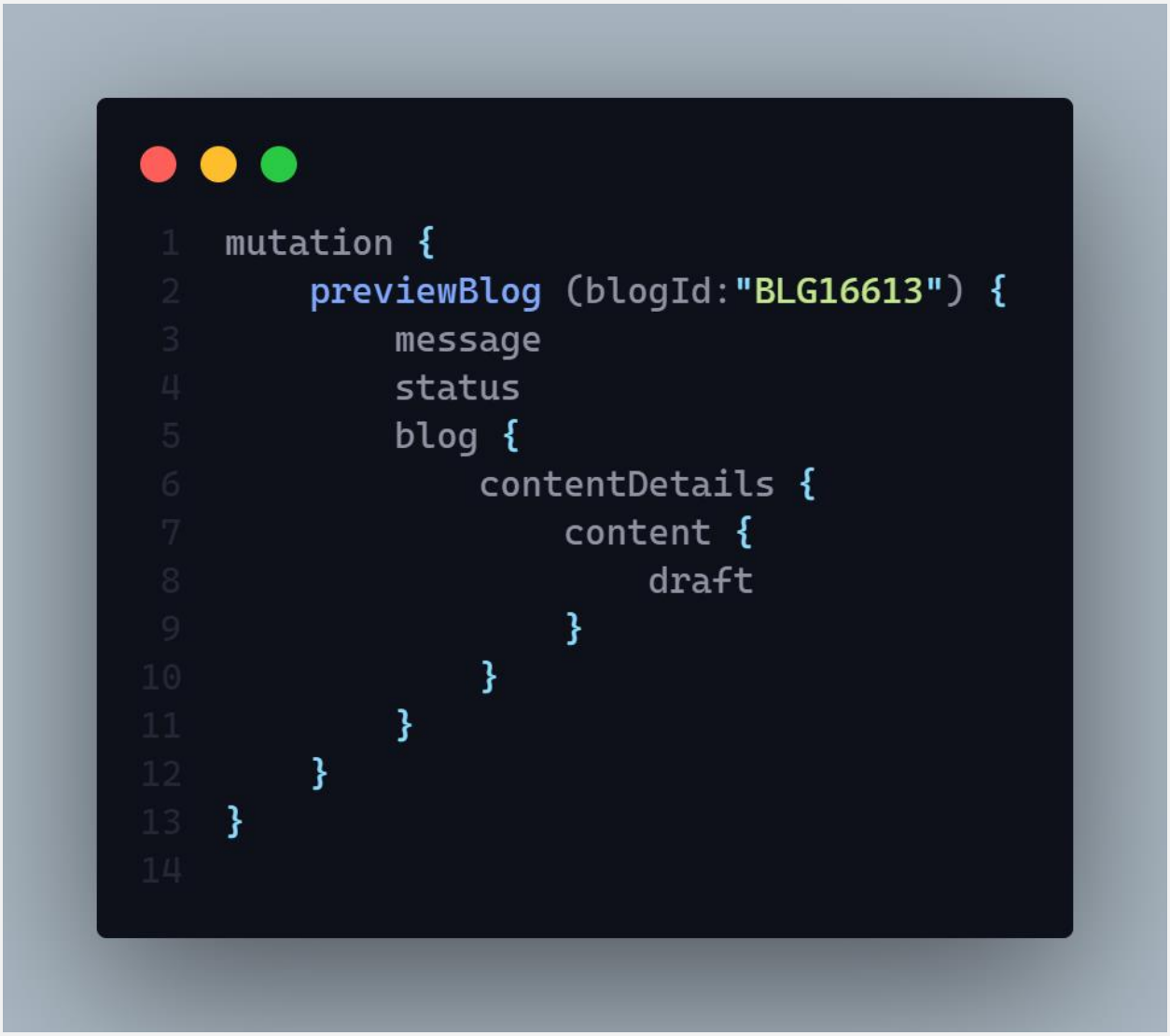
```
{
    "data": {
        "previewBlog": {
            "message": "Blog not found",
            "status": 404,
            "blog": null
        }
    }
}
```

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:


mutation {
previewBlog (blogId:"BLG16613") {
message
status
blog {
contentDetails {
content {
draft
}
}
}
}
}

11. Delete an individual blog by slug

API: localhost:3030/blog

Type: GraphQL

Method : POST

Access: **:** client, admin

Auth: true

Body :

```
mutation {
    deleteBlog (slug: "Second_Blog_web_development_web_web3.0_1643495385820")
```

Params: N/A

Query: N/A

Data: N/A

Hints: N/A

Response:


```
{
    "data": {
        "deleteBlog": {
            "message": "Blog deleted successfully",
            "status": 202
        }
    }
}
```

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**

API: localhost: 3030/blog

Type: GraphQL

Method : POST

Access: **:** client

Body :

```
mutation {
    updateBlog (
        slug: "Second_Blog_web_development_web_web3.0_1643495385820",
        input: {
            subCategory: ["React", "web",
            "web3.0"]
            mainCategory: "Web Design",
            keyWord: [ "web",
            "development", "react"]
        }
    )
```

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


mutation {
updateBlog (
slug: "Second_Blog_web_development_web_web3.0_1643495385820",
input: {
subCategory: ["React", "web",
"web3.0"]
mainCategory: "Web Design",
keyWord: ["web",
"development", "react"]
}
) {
message
status
}
}

13. Update blog title or cover picture by slug

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

API: localhost:3030/blog

Type: GraphQL

Method : POST

Access: **:** client

Body :

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

```
mutation {
    updateBlogImage (
        input: {
            base64: "",
            size: 25458
        }
        slug: "Second_Blog_web_development_web_web3.0_1643495385820"
        type: "cover" or "title"
    )
```

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:

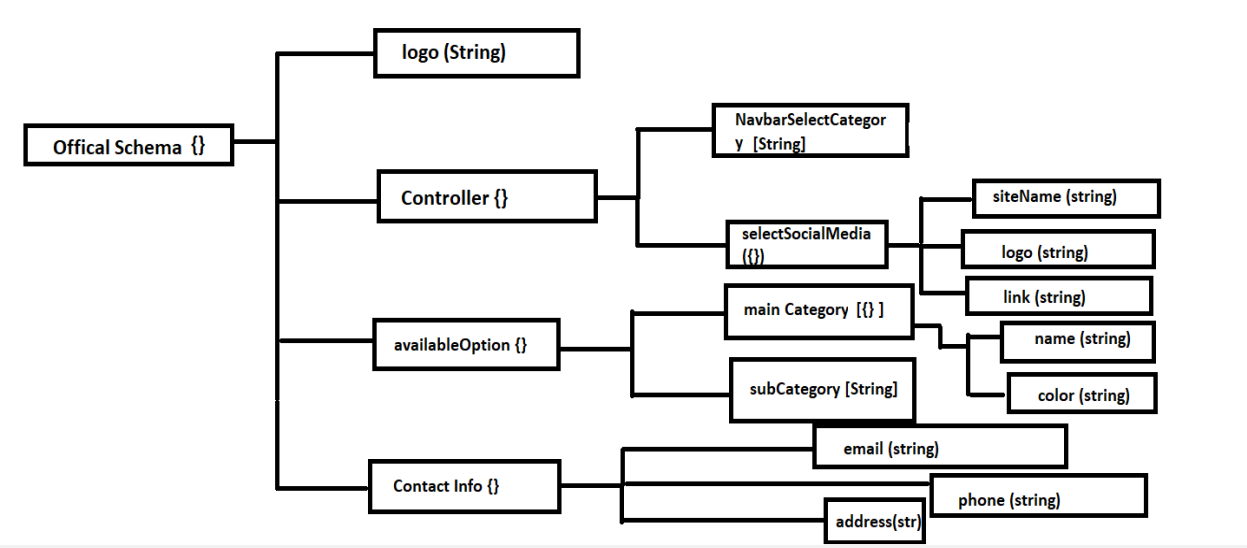
Description: If status code is

- **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

```
mutation {
    updateBlogImage (
        input: {
            base64: "",
            size: 25458
        }
        slug: "Second_Blog_web_development_web_web3.0_1643495385820"
        type: "cover" or "title"
    ) {
        message
        status
    }
}
```

Official Schema



1. Create an official schema

API: localhost:3030/official

Type: GraphQL

Method : Mutation/ POST

Access: **:** admin

IsAuth : true

Body : In the body part it need mandatory some data =>

1. email => //mandatory
2. companyName =>//mandatory
3. logo
4. phone => //mandatory
5. address => //mandatory
6. availableSocialMedia => [] => {} =>
    a. siteName => //mandatory
    b. logo => //mandatory
7. available MainCategory => [] => {} =>
    a. name => //mandatory
    b. color => //mandatory
8. availableSubCategory => [] => String //mandatory

```
input LogoInput {
        base64: String
        size: Int
    }
    input availableSocialMediaInput {
        siteName: String!
        logo:String!
    }
    input availableMainCategoryInput {
        name: String!
        color:String!
    }
    input createOfficialInput {
        email: String!
        companyName: String!
        logo: LogoInput
        phone: String!
        address: String!
        availableSocialMedia: [availableSocialMediaInput]!
        availableMainCategory: [availableMainCategoryInput]!
        availableSubCategory: [String]!
    }
```

Params: N/A

Query: N/A

Data: N/A

Hints: It will create a schema but only one schema will be create in the entire website.

Response:

I

Description: if status code is

- **201 =>**
  - Official data created successfully

- **406 =>**
  - Official schema available you can not create multiple official
  - Only jpg jpeg png format are accepted
  - Logo upload failed
  - Official data creation failed
  - Any run time error
- **401** =>
  - Permission deniend
  - Unauthorized user
- 403 =>
  - Schema Vaidation error message

Workflow: This api will take some necessary information of the company and create a new official company schema. But condition is there don't allow any multiple official schema.

Demo :

```
mutation {
    createOfficialData (data: {
        email: "xyz@gmail.com",
        companyName: "Blogger",
        phone: "01521427881",
        logo: {
            base64 : "",
            size: ""
        },
        address: "Plot 16 Aftab Uddin Ahmed Rd, Dhaka 1229",
        availableMainCategory: [
            {
                name: "Web Developnment",
                color: "#ddddd"
            }
        ],
        availableSocialMedia: [
            {
                siteName: "facebook",
                logo: "fa-brands fa-facebook-f"
            }
        ],
        availableSubCategory: [
            "random",
            "html-design"
        ]
    }) {
        message,
        status,
        data {
            companyName
            logo,
            controller {
                navbarSelectCategory
            }
            availableOption {
                mainCategory {
                    name,
                    color
                }
                socialMedia {
                    logo
                    siteName
                }
                subCategory
            }
        }
    }
}
```

2. Add available mainCategory with color or available sub category (gql)

API: localhost: 3030/official

Type: GraphQL

Method : Mutation / POST

Access: **:** admin

IsAuth: true

Body : In the body part we need two types of data

mainCategory => [] => {} =>

1. name
2. color

OR

subCategory => [] => any String. Example : [“Design”]

1. User can pass only mainCategory or only sub category.
2. User can not pass both
3. User have to wrap all in a object

```
addMainOrSubCategory (data: {
        mainCategory : [
            {
                name: "Web Design"
                color: "#000"
            },
            {
                name: "Web Developnment"
                color: "#5rrrr"
            }
        ]
        // OR
        subCategory : ["React", "Next js"]
    })
```

Params: N/A

Query: N/A

Data: N/A

Hints: N/A

Response:

```
type addMainOrSubCategoryResponse {
        message: String!
        status: Int!
    }
```

Description:

if status code is

- **202 =>**
  - Successfully operation done
- **406 =>**
  - Failed operation
  - Any runtime error
- **401** =>
  - Permission deniend
  - Unauthorized user
- **404 =>**
  - No official info found

WorkFlow:

With this API =>

1. User can add new main category which will available for use in entire website. Which contain a category name and also a default color hexa code. Example : for white color #dddd or dddd. Here main category will be a array and name, color will be a object property of that mainCategory array.

2. User can add new sub category. Here user need to put a new name in a array. This value must be string
3. User can update main category item’s default color name. Example if color was red it can update is to white but condition is it needs to get the name of that particular main category name.

3. Add new contact social media to contact officially or update existing contact social media link to contact officially (gql)

API: localhost:3030/official

Type: GraphQL

Method : MUTATION / POST

Access: **: admin**

IsAuth: true

Body : In the body part it will receive a array of object where it contains

[] => {} =>

1. siteName (ex: facebook)
2. link (https://www.facebook.com/)

```
addOrUpdateContactSocialMedia (socialMedia: [
        {
            siteName: "facebook",
            link: "https://www.facebook.com/"
        },
        {
            siteName: "youtube",
            link: "https://www.youtube.com/watch?v=MKP6p7nXTVc"
        },
    ])
```

Params: N/A

Query: N/A

Data: N/A

Hints: N/A

Response:

Description:

if status code is

- **202 =>**
  - Successfully operation done

- **406 =>**
  - Failed operation
  - Any runtime error
- **401** =>
  - Permission deniend
  - Unauthorized user
- **404 =>**
  - No official info found

Demo:

```
mutation {
    addOrUpdateContactSocialMedia (socialMedia: [
        {
            siteName: "facebook",
            link: "https://www.facebook.com/"
        },
        {
            siteName: "youtube",
            link: "https://www.youtube.com/watch?v=MKP6p7nXTVc"
        },
    ]) {
        message
        status
    }
}
```

WorkFlow:

With this API =>

1. User can add a new social media link for contact to the company where it contains a site name including site link. It will be sent as a array object
2. User can also update exist company social media link by using the social media name

4. Select main category or update for navbar (gql)

API: localhost:3030/ official

Type: GraphQL

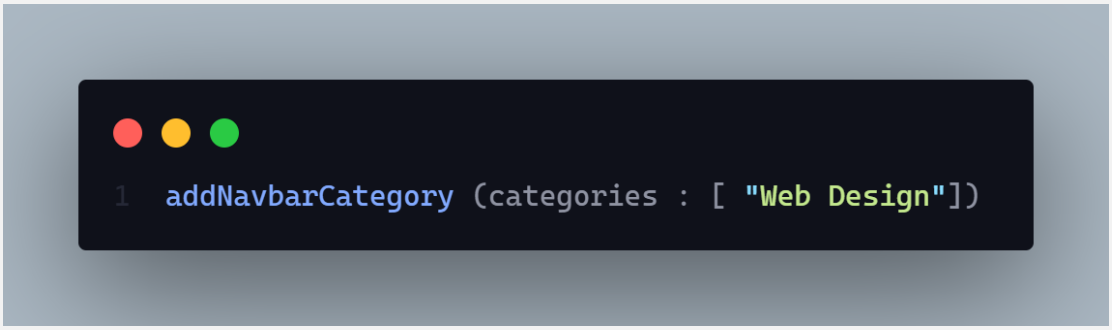
Method : MUTATION/ POST

Access: **:** admin

IsAuth: true

Body : It will take a array of category

[] => category name



Params: N/A

Query: N/A

Data: N/A

Hints: N/A

Response:

Description:

if status code is

- **202 =>**
  - Successfully operation done
- **406 =>**
  - Failed operation
  - Any runtime error
- **401** =>
  - Permission deniend
  - Unauthorized user
- **404 =>**
  - No official info found

WorkFlow:

With this API =>

1. User can add new navbar visible category and also can update that array of navbar visible category

5. Get all official info –

1. **give all available main category**
2. **give all available sub category**
3. **give all available social media for contact**
4. **If no query have passed then give all official data.**

API: localhost:3030/ official

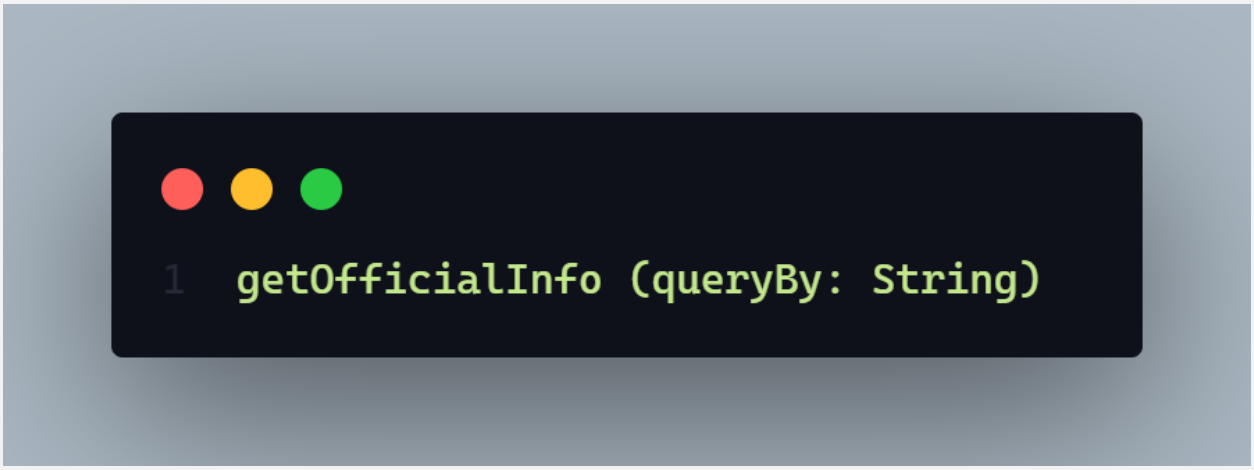
Type: GraphQL

Method : MUTATION / POST

Access: **:** ALL

Body : To get

1. All available sub category you need to input => queryBy = “subCategory”
2. All available main category you need to input => queryBy = “mainCategory”
3. All available social media you need to input => queryBy = “socialMedia”
4. If no input has been pass in query by field then it will give all information of official information



Here value queryBy will be →

1. subCategory
2. mainCategory
3. socialMedia

Params: N/A

Query: N/A

Data: N/A

Hints: N/A

Response:

```
type getOfficialInfoResponse {
        message: String!
        status: Int!
        info : Official
    }
```

Description:

if status code is

- **202 =>**
  - Info found
- **406 =>**
  - Any runtime error
- **401** =>
  - Unauthorized user
- **404 =>**
  - Official info not found

WorkFlow:

With this API =>

1. User can get all official info if query by input will give as a empty string
2. User can get only available main category of official if user give query by is “mainCategory”
3. User can get only available sub category of official if user give query by is “subCategory”
4. User can get only available social media of official if user give query by is “socialMedia”

Demo:

1. **If User want to get all available Social media option in the website**

```
//if user want to get all available social media of official info
query {
    getOfficialInfo (queryBy: "socialMedia") {
        message
        status
        info {
            availableOption {
                socialMedia {
                    siteName
                    logo
                }
            }
        }
    }
}
```

2. **If User want to get all available main category option in the website**

```
query {
    getOfficialInfo (queryBy: "mainCategory") {
        message
        status
        info {
            availableOption {
                mainCategory {
                    name
                    color
                }
            }
        }
    }
}
```

3. **If User want to get all available sub category option in the website**

```
query {
    getOfficialInfo (queryBy: "mainCategory") {
        message
        status
        info {
            availableOption {
                subCategory
            }
        }
    }
}
```

Otherwise it will show all data of official info

```
query {
    getOfficialInfo (queryBy: "") {
        message
        status
        info {
            controller {
                navbarSelectCategory
                selectSocialMedia {
                    siteName
                    link
                }
            }
            availableOption {
               subCategory
            }
            companyName
            logo
            contactInfo {
                email
                phone
                address
            }
        }
    }
}
```

6. Upload or update logo

API: localhost:3030/ official

Type: GraphQL

Method : POST/MUTATION

Access: **: admin**

Body : it will take only a logo base 64 data with the size

Logo => {} =>

Base64 => String (mandatory)

Size => Number (mandatory)

Params: N/A

Query: N/A

Data: N/A

Hints: N/A

Response:

Description:

if status code is

- **202 =>**
  - Logo upload successfully
- **406 =>**
  - Any runtime error
  - Logo update failed
  - Logo upload failed
- **401** =>
  - Unauthorized user
  - Permission deneid

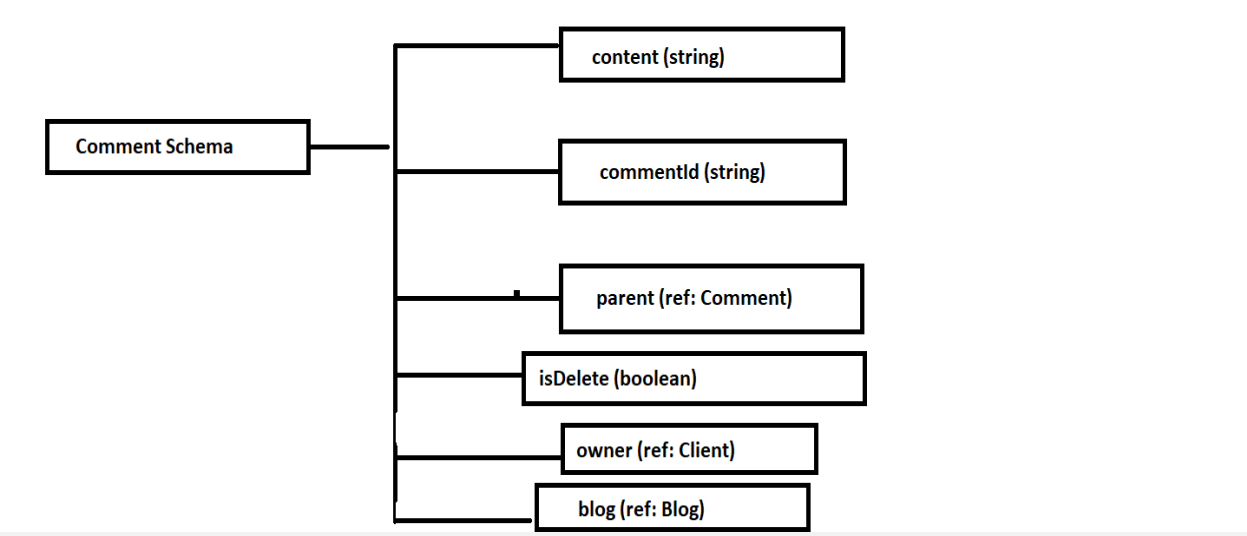
WorkFlow:

With this API =>

1. User can upload or updated company logo
2. Only Admin can access this route

Demo:

Comment Schema



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

API: localhost:3030/comment

Type: GraphQL

Method : MUTATION / POST

Access: **:** client

IsAuth: true

Body : It will take =>

1. content (required)
2. parent (it will give parent unique comment id)
3. blog (it will give blog's unique object id which provided by the mongodb database) (required)

```
publishComment (input: {
        content: "I am from comment 2 child one child two",
        parent: "CMNT34518",
        blog: "61f5bfd93ad6fd3d2214bf38"
    })
```

Params: N/A

Query: N/A

Data: N/A

Hints: N/A

Response:

```
type publishCommentResponse {
        message: String!
        status: Int!
    }
```

Description:

if status code is

- **201 =>**
  - Comment successfully published
- **406 =>**
  - Any runtime error
  - Joi Validation error
  - Comment published failed
- **401** =>
  - Unauthorized user
  - Permission deneid

WorkFlow:

With this API =>

1. User published a new comment
2. It can published child comment which actually known as reply

Demo:

```
mutation {
    publishComment (input: {
        content: "I am from comment 2 child one child two",
        parent: "CMNT34518",
        blog: "61f5bfd93ad6fd3d2214bf38"
    }) {
        message
    }
}
```

2. Update comment by coment id

API: localhost:3030/comment

Type: GraphQL

Method : MUTATION/POST

IsAuth: false

Access: **:** client

Body : it will take =>

1.content (it will contain updated comment content) (required)

2. commentId (it will contain comment unique id) (required)

```
modifiedComment (
        content: "I am from comment 1",
        commentId: "CMNT18151"
    )
```

Params: N/A

Query: N/A

Data: N/A

Hints: N/A

Response:

Description:

if status code is

- **202 =>**
  - Comment has been modified
- **406 =>**
  - Any runtime error
- **401** =>
  - Unauthorized user
  - Permission denied
- **403 =>**
  - Comment update failed
- **404 =>**
  - Comment id required

WorkFlow:

With this API =>

1. User can update existing comment by using the unique comment id what is auto generated during the published time

3. Get all comment by blog id name

API: localhost:3030/comment/get/all/:blogId

Type: REST

Method : GET

Access: **: all**

Body : N/A

Params: blog object id =>

1. This blog object id is provided by the mongodb database

Query: N/A

Data: N/A

Hints: N/A

Response:

```
res.json ({
                    message: `${getAllComment.length} ${getAllComment.length > 1 ? "comments" : "comment"} found`,
                    comment: getAllComment,
                    status: 202
                })
```

Description:

if status code is

- **202 =>**
  - N comment found

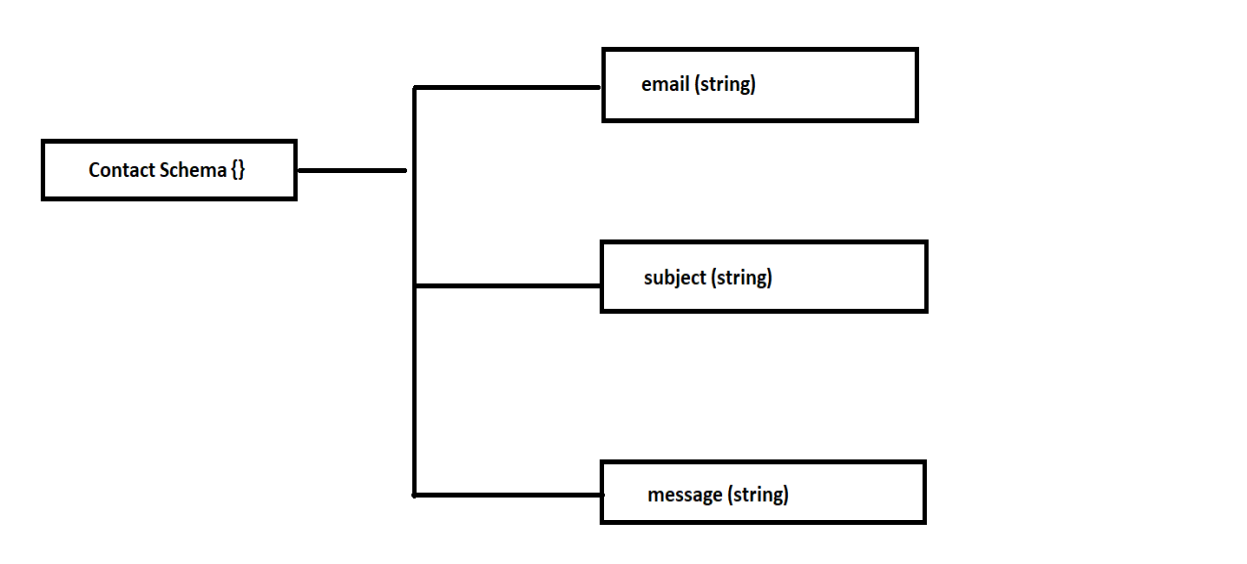
  406 =>
  - Any runtime error
- **404 =>**
  - No comment found
  - Blog id requried

WorkFlow:

With this API =>

1. User can see all comment of a particular blog
2. As a response it will give all comment details including

a. Owners =>
    i. `personalInfo`
        1. `address`
            a. `district`
            b. `division`
            c. `country`
        2. firstName
        3. lastName
        4. email
        5. contactNumber
        6. profilePicture
        7. coverPicture
        8. gender
        9. bio
    ii. _id
b. Blogs =>
    i. `contentDetails`
        1. `content`
            a. draft
            b. published
        2. subCategory
        3. keyword
        4. title
        5. mainCategory
        6. coverPic
        7. titlePic
        8. _id



1. Create a new contact

API: localhost:3030/contact

Type: GraphQL

Method : MUTATION/POST

Access: **:** all

IsAuth: false

Body : it will take =>

1. subject => String // mandatory field
2. email => String //mandatory // it have to a valid email
3. message => String //mandatory field

```
createNewContact (input: {
        subject: "Random",
        email: "sadmanishopnil@gmail.com",
        message: "Hello I am from message one. This is my first message"
    })
```

Params: N/A

Query: N/A

Data: N/A

Hints: N/A

Response:

Description:

if status code is

- **201 =>**
  - Message sent successfully

  406 =>
  - Any runtime error
- **403 =>**
  - Message send failed

WorkFlow:

With this API =>

1. User can contact with any admin authority and tell the problem

Demo:

```
mutation {
    createNewContact (input: {
        subject: "Random",
        email: "sadmanishopnil@gmail.com",
        message: "Hello I am from message one. This is my first message"
    }) {
        message
        status
        contact {
            email
            isRead
            createdAt
            updatedAt
            isRead
            subject
            message
            contactId
        }
    }
}
```

2. Get all contact

Query by =>

1. **Read one**
2. **Uread one**

Sort by =>

1. **Ascending order by email**
2. **Descending order by email**
3. **Ascending order by date**
4. **Descending order by date**

API: localhost:3030/contact

Type: GraphQL

Method : QUERY/POST

Access: **:** admin

Body : It will take two data =>

1.queryBy =>
option =>

a. read => it will query all read contact message

b. unread => it will query all unread contact message

2. sortBy =>

a. ascendingByDate => It will sort all contact message in a ascending order by the create date

b. descendingByDate => it will sort all contact message in a descending order by the created date

c. ascendingByEmail => it will sort all contact message in a ascending order by the sender email

d. descendingByEmail => it will sort all contact message in a descending order b the sender email

```
showAllContact (input: {
    queryBy: "unread" //expected data ⇒ unread, read
    sortBy: "descendingByDate" //expected data ⇒ ascendingByDate, descendingByDate, ascendingByEmail,descendingByEmail
})
```

Params: N/A

IsAuth: true

Query: N/A

Data: N/A

Hints: N/A

Response:

Description:

if status code is

- **202 =>**
  - N contacts found //here N can be any value (0 – 9)

  406 =>
  - Any runtime error
- **404 =>**
  - No contact found

WorkFlow:

With this API =>

1. Admin can find all contact
    a. Filter by read and unread category
    b. Sort by ascendingByDate , descendingByDate, ascendingByEmail, descendingByEmail order

Demo:

```
query {
    showAllContact (input: {
        queryBy: "unread"
        sortBy: "descendingByDate"
    }) {
        message
        status
        contact {
            email
            isRead
            createdAt
            updatedAt
            isRead
            subject
            message
            contactId
        }
    }
}
```

3. Get individual contact by id

API: localhost:3030/contact

Type: GraphQL

Method : QUERY/POST

Access: **:** admin

Body : it will take one data =>

1. contactId => It is a auto generate contact id.

Params: N/A

IsAuth: true

Query: N/A

Data: N/A

Hints: N/A

Response:

Description:

if status code is

- **202 =>**
  - Contact found

  406 =>
  - Any runtime error
- **404 =>**
  - Contact not found
- **401** =>
  - Unauthorized user
  - Permission denied

WorkFlow:

With this API =>

1. Admin can get access of individual contact information.

Demo:

```
query {
    showIndividualContact (contactId: "CNT52789") {
        message
        status
        contact {
            email
            isRead
            createdAt
            updatedAt
            isRead
            subject
            message
            contactId
        }
    }
}
```

4. Reply to a contact message by provided email

API: localhost:3030/contact

Type: GraphQL

Method : POST/ MUTATION

Access: **:** admin

IsAuth: true

Body : it will take two data =>

1.contactId => It is a auto generate contactId

2. reply => What content admin want to reply to the sender

Params: N/A

Query: N/A

Data: N/A

Hints: N/A

Response:

Description:

if status code is

- **202 =>**
  - Reply has been successfully sent

  406 =>
  - Any runtime error
- **403=>**
  - Message sent but contact status is currently unread
  - Reply sent falied
- **401** =>
  - Unauthorized user
  - Permission denied
- **404=>**
  - Contact not available please check in read section

WorkFlow:

With this API =>

1. Admin can sent a reply to the sender in the sender provided email

Demo :


mutation {
sendReply (
contactId: "CNT52789",
reply: "Thanks for your response"
) {
message
status
}
}