Database Info

ID: 4
Name: login
Admin: 11
Sub Admins: None
Members: None

Tables

Table Name: users

Column Name	Datatype	Constraint	Key
name	VARCHAR	NOT NULL	NONE
pass	VARCHAR	NOT NULL	NONE
mail	VARCHAR	NOT NULL	NONE

Dynamic Add Data Example:

```
let data = {
  db_id: "4",
  tables: [
    {
      table_name: "users",
      data: [
        { column_name: "name", data_to_insert: "VALUE", userid },
        { column_name: "pass", data_to_insert: "VALUE", userid },
        { column_name: "mail", data_to_insert: "VALUE", userid }
      ]
    }
  ]
}
```

Dynamic Find Example:

```
let data = {
  db_id: "4",
  tables: [
    { table_name: "users", unique: userid }
  ]
}
```

Dynamic Update Example:

```
let data = {
```

```
  db_id: "4",
  tables: [
    {
      table_name: "users",
      unique: userid,
      data: [
        { column_name: "name", data_to_insert: "NEW_VALUE", userid },
        { column_name: "pass", data_to_insert: "NEW_VALUE", userid },
        { column_name: "mail", data_to_insert: "NEW_VALUE", userid }
      ]
    }
  ]
}
```

JWT Token Authentication Documentation

This system uses JWT token authentication. A token is given after login when the user navigates to the home page.
To perform any CRUD operations, you will need the **userid**. To get it, send a request to: `http://localhost:3001/get_id`

Adding Data to Database

Send the following variable structure to the GraphQL endpoint:

```
let data = {
  db_id: "given in documentation",
  tables: [
    {
      table_name: "table name from EHR data to operate on",
      data: [
        { column_name: "column1", data_to_insert: "value1", userid },
        { column_name: "column2", data_to_insert: "value2", userid },
        { column_name: "column3", data_to_insert: "value3", userid }
      ]
    }
  ]
}
```

Fetch example:

```
fetch("http://localhost:3001/graphql", {
  method: "POST",
  credentials: "include",
  headers: {
    "Content-Type": "application/json",
    "Authorization": `Bearer ${token}`
```

```
  },
  body: JSON.stringify({
    query: `
      mutation add_row($data: JSON!) {
        add_row(data: $data) {
          id
          db_name
          data
        }
      }
    `,
    variables: { data }
  })
})
.then(res => res.json())
.then(data => console.log(data))
.catch(err => console.error(err));
```

Finding Data

To find any row in any table, send:

```
let data = {
  db_id: "given in documentation",
  tables: [
    { table_name: "table1", unique: userid },
    { table_name: "table2", unique: userid }
  ]
}

fetch("http://localhost:3001/graphql", {
  method: "POST",
  credentials: "include",
  headers: {
    "Content-Type": "application/json",
    "Authorization": `Bearer ${token}`
  },
  body: JSON.stringify({
    query: `
      mutation find($data: JSON!) {
        find(data: $data)
      }
    `,
    variables: { data }
  })
})
.then(res => res.json())
.then(data => console.log(data))
.catch(err => console.error(err));
```

Updating Data

To update any row in any table, use:

To update any row in any table, use:

```
let data = {
  db_id: "given in documentation",
  tables: [
    {
      table_name: "table1",
      unique: userid,
      data: [{ column_name: "col1", data_to_insert: "xyz", userid }]
    },
    {
      table_name: "table2",
      unique: userid,
      data: [{ column_name: "col2", data_to_insert: "xyz", userid }]
    }
  ]
}

fetch("http://localhost:3001/graphql", {
  method: "POST",
  credentials: "include",
  headers: {
    "Content-Type": "application/json",
    "Authorization": `Bearer ${token}`
  },
  body: JSON.stringify({
    query: `
      mutation update($data: JSON!) {
        update(data: $data)
      }
    `,
    variables: { data }
  })
})
.then(res => res.json())
.then(data => console.log(data))
.catch(err => console.error(err));
```