Routes:

An error response will always be of the form:
{"success": False, "error": message}
The following routes assume an api call goes through successfully.

1. POST /api/signin/

Request: `
{
"username": username <string>,
"password": password <string>,
“first_name”: firstName <string>,
“last_name”: lastName <string>
}
Response:
{
"success": True,
"data": {
“user_id": userId <integer, the id associated with this user>,
"username": username <string, the username of this user>,
“first_name”: firstName <string>,
“last_name”: lastName <string>
}
}

Eg: Request json is going to look like
{“username”: “ab123”, “password”: “pwd”, “first_name”: “Jack”, “last_name”: “Smith”}
Response is pretty straightforward and is not really important because we won’t be using any of these further. The “data” would look something like
{“user_id”: 1, “username”: ab123, “first_name”: “Jack”, “last_name”: “Smith”}

2. POST /api/login/

Request:
{
"username": username <string>,
"password": password <string>
}
Response:
{
"success": True,
"data": {
“user_id": userId <integer, id of this user>,
"username": username <string, username of this user>,
"token": access_token <string, unique token associated with this user>
}
}

Calling this is exactly the same as just using the username and password fields of the signing route, except this time it checks if the user actually has an account in the database, an error is reported if the account doesnt already exist. For the response json, you can access the “data” element, within which you can access the “Token” element, and this needs to be stored in a global var probably because we use it throughout the app, I will refer to this variable as simply ‘token’ from here, it basically just gets called in the header for every subsequent route.
To log in the user we just created an account for, request json:
{“username”: “ab123”, “password”: “pwd”}
And the response would be something like “data”:
{“user_id”: 1, “username”: “ab123”, “token”: “u2i34y3927ry823rgyu22983y48u32”}
This string for token gets stored in the token variable.

3. POST /api/newChat/

Request:

```
{
"from": fr_id <integer, the userId of the user sending the message>,
"to": to_id, <integer, the userId of the user receiving the message>
}, { headers: { Authorization: "Bearer " + token } }
```

Response:

```
{
"success": True,
"data": {
  "from": fr_id,
  "to": to_id,
  "from_channel": fr_channel,
  "to_channel": to_channel,
  "channel_name": chat_channel,
}
}
```

Suppose we have another user: {“user_id”: 2, “username”: “jk345”, … } This route checks if a channel between these two users already exists, and if it does not creates it. The request is going to look like
{“from”: 1,”to”: 2},
{headers: {Authorization: "Bearer " + “u2i34y3927ry823rgyu22983y48u32”}}
if we want to send a msg from ab123 to jk345.

The response is going to look like (the “data” part at least)
{“from”: 1, “to”: 2, “from_channel”: “private_chat_1”,
“to_channel”: “private_chat_2”, “channel_name”: “private_chat_1_2”}
These are the channels that we subscribe to.

4. POST /api/authentication/ (don’t need to worry about this one atm)

Request:

```
{
```

```
"channel_name": channel_name,
"socket_id": socket_id
}, { headers: { Authorization: "Bearer " + token } }
```

5. POST /api/sendMsg/

   Request:

```
{
"From": from_user_id,
"To": to_user_id,
"message": message_content,
"channel_id": channel_id
}, { headers: { Authorization: "Bearer " + token } }
```

   Response:

```
{
"success": True,
"data": {
  "Message id": id,
  "Message contents": contents,
  "From": fr,
  "To": to,
  "Sent in channel": channel_id
      }
}
```

6. GET /api/getMsg/<int:channel_id>

   Request:

```
{ headers: { Authorization: "Bearer " + token } }
```

   Response:

```
{
"success": True,
"data": [
  {
      "Message id": id,
      "Message contents": contents,
      "From": fr,
      "To": to,
      "Sent in channel": channel_id
  },
  ... and so on for all the messages in this channel, data would be a list
in this case
]
}
```

7. GET /api/getAllUsers/

   Request:

```
{ headers: { Authorization: "Bearer " + token } }
```

Response:

```
{
"success": True,
"data": [
  {
      "User id": self.id,
      "Username": self.username
  },
… and so on for all users in the database, data would be a list in this
case
]
}
```