Base: localhost:3000/BackEnd/

Error protocol, if an error occurs

- Error:-1 - UserInput with maybe an err in the json explaining the error, if needed
- Error: -2 - server Error

Create User

Path: createUser/

Method: POST

Body:

- *UserName* (String) - username
- *Password* (String) - password
- *Birthday (string) -(“mm/dd/yyyy”)*
- *Gender (String)- “M”,”F”, “MF”*
- *GenderInto - “M”, “F”, “MF”*

Returns:

- {UserID:int,
- UserName:varchar(30),
- Picture:varchar(30)/null,
- Birthday_year int Not Null,
- Birthday_month int Not Null,
- Birthday_day int Not Null,
- Gender:varchar(2),
- GenderInto:varchar(2),
- loc: varchar(45) null,
- InARelationship:boolean,
- minAge: int,null,
- maxAge:int, null }
- Or if error
- err = “User Name already taken”

Login

Birthday_year int Not Null, Birthday_month int Not Null, Birthday_dayPath: login/

Method: POST

Body:

- *UserName* (String) - username
- *Password* (String) - password

Returns:

- {UserID:int,
- UserName:varchar(30),
- Picture:varchar(30)/null,

- Birthday_year int Not Null,
- Birthday_month int Not Null,
- Birthday_day int Not Null,
- Gender:varchar(2),
- GenderInto:varchar(2),
- loc: varchar(45) null,
- InARelationship:boolean,
- minAge: int,null,
- maxAge:int, null }

EditPassword
Path: editPassword/
Method: POST
Body:
- *UserID* (int) - id
- *oldPassword* (String) - password
- *newPassword* (String) - first name

Returns:
- Success:0

Add user Pref
Path: addUserPref/
Method: POST
Body:
- *UserID* (int) - id
- Name (String) - Name of Pref

Returns:
- Success:0

Get user Pref
Path: addUserPref/
Method: POST
Body:
- *UserID* (int) - id

Returns:
- {InterestID (int) - id,
- Name varchar(45) - name of Pref,
- Description varchar(200), Null - Desc of Pref}

getPrefs
Path: getPrefs/
Method: POST
Body:

- *nothing*

Returns:

- {InterestID (int) - id,
- Name varchar(45) - name of Pref,
- Description varchar(200), Null - Desc of Pref}

Set Age - this will set the min and max Ages for matching
Path: setAge/
Method: POST
Body:

- *UserID* (int) - id
- *minAge*(int) - min Age person wants
- *maxAge* (int) - max Age person wants

Returns:

- Success:0

Block User- this will block a match from showing up or being able to communicate with you
Path: blockUser/
Method: POST
Body:

- *UserID1* (int) - id of User who is blocking
- *UserID2* (int) - id of User who the user wants to block

Returns:

- Success:0

Get Matches
Path: getMatches/
Method: GET
Body:

- *UserID1* (int) - id of User who of user who is checking matches

Returns:An array of JSON like this,
//TODO change like this

- [{ UserID: int -first userID,
- UserName string- name of User matched}]

Get Messages
Path: getMessages/
Method: GET
Body:
- *UserID1* (int) - id of one of the two users getting Messages for
- *UserID2* (int) - id of one of the two users getting Messages for

Returns: An array of JSON like this,
- {"MessageID":(int) - ID given to make each message unique and know time order,
- "UserID1":(int) - sender,
- "UserID2":(int) - receiver ,
- "Message_Title":(string)null,
- "Message":"Hello",
- "UserID1_Read":boolean,
- "UserID2_Read":boolean}

Get UserLanguages
Path: getUserLanguage/
Method: GET
Body:
- *UserID* (int) - id of user

Returns: An array of JSON like this,
- {Name (String)- not Null - name of Language}

Add User Language
Path: addUserLanguage/
Method: POST
Body:
- *UserID* (int) - id of User who is blocking
- *Name(string) - Name of Language.*

Returns:
- Success:0

EditPicture

Path: editPicture/
Method: POST
Body:

- *UserID* (int) - id
- *Picture* (String) - link

Returns:

- Success:0

Socket IO section send over http://localhost:3000/

What the server Emits

Message: if user receives a message:

Sends {UserID (int) - id of User sending the message,
Message (string):message being sent to client}

Success: if last message sent was a success it will send this

Sends{success:0}

err: If last message sent resulted in an error

Sends{err: -1 or -2}
-1 means that the user does not exist, you are not matched with the user or you have been blocked from said user
-2 means a server side issue with your request, see log, in server for issue

What the server expects the client to emit

hello- call right after connect to set up connection for other users to send

Expects

{UserID:id of current user}

send - call that allows you to send a message

expects
{UserID- the userid of the user you would like to send the message to,
Message- the message that you would like to send the user}