Online Chatting Application v1.0

Created by John Owens

Summary
The Online Chatting Application (O.C.A.) is a RESTful web application created using the DropWizard framework and the Java programming Language. The web application allows for users to create accounts in order to write on an online chat log. Users will also be able to update their accounts, delete their accounts and get information about their accounts. User will also be able to write messages in a chat log, see previous messages and update and/or delete previous messages.

Endpoints of the web application are documented below

Login to Account

POST - http://localhost:8080/account/login

Description

This request will log a user in successfully if their account was successfully created beforehand and added to the database.

JSON Body Example

```
{
  "user": "foobar",
  "pass": "foobar",
}
```

Admin Login - `"user": "admin","pass": "adminPassword"`

Responses

201 Created

```
{
    "code": 201,
  "response": "Successfully Logged In"
}
```

404 Not Found

Credentials do not match

LIST ALL ACCOUNTS - Only Admin Can Do This

Get - http://localhost:8080/account

Description

If you are signed in as the admin, this will give you a list of all users and their account information. Normally, I would never give the admin credentials on a requirements document, but since this is a graded assignment where testing is required, then refer to the red text in the previous login endpoint.

Responses

201 Created

Created

```
{
   "username": "foobar",
  "password": “foobar",
  "age": “foobar”,
  "gender": “foobar”
} … (etc., list all accounts)
```

401 Unauthorized

You are not an admin

401 Unauthorized

You are not logged in

Get Specific Account

Get -   http://localhost:8080/account/specific

Description

If you are signed in, you will be able to see your account information.

Responses

201 **Created**

Created

```
{
    "code": 201,
  "response": “Account Made: true"
}
```

401 **Unauthorized**

You are not logged in

Delete Your Own Account

```
Delete -   http://localhost:8080/account
```

Description

If you are signed in, you will be able to delete your account from the database.

Responses

201 Created

Created

```
{
    "code": "201",
  "response": “Username ‘foobar’ Deleted”
}
```

401 Unauthorized

You are not logged in

Add Account to Database

Put - http://localhost:8080/account

Description

If you are not signed in, you will be able to create an account.

JSON Body Example

```
{
  "user": "foobar",
  "pass": "foobar",
  "age": "foobar",
  "gender": "foobar"
}
```

Responses

201 Created
Created

```
{
     "code": "201",
  "response": “Account Made: true”
}
```

404 Not Found
Account Already Exists

406 Not Acceptable
Must Log out to make Account

Edit Account Username or Password

Post - http://localhost:8080/account

Description

If you are signed in, you will be able to update your username and/or password.

JSON Body Example

```
{
  "user": "foobar2",
  "pass": "foobar2"
}
```

Responses

201 Created
Created

```
{
    "code": 201,
  "response": "Account username 'foobar' and
password 'foobar' changed to username 'foobar2'
and password 'foobar2'"
}
```

401 Unauthorized
You are Not Logged In

Log out of Account

Delete - http://localhost:8080/account/logOut

Description

This request will allow the user to sign out of their account and forfeit their authentication token being valid.

Responses

202 Created

Created

```
{
    "code": 202,
  "response": “Successfully Logged Out”
}
```

401 Unauthorized

Cannot Log Out if not Logged In

Show All Messages In Chat

Get - http://localhost:8080/chat

Description

This request will allow the user to see all the messages posted in the chat database by all members.

Responses

202 Created

Created

```
{
    "time": "foobar",
   "user_name": "foobar",
   "message": "Lorem Ipsum",
   "id": "foobar"
} ... (etc., list all messages)
```

401 Unauthorized

You are not logged in

Delete Message At Specific Time

Delete - http://localhost:8080/chat/time/{foobar}

Description

This request will allow the user to delete a message, if they have both the time they sent the message (in database) and if they themselves wrote the message.

Responses

201 Created

Created

```
{
   "code": 201,
  "response": “Message at ‘foobar’ Deleted"
}
```

404 Not Found

No such time in database exists

401 Unauthorized

Cannot delete someone else’s message

401 Unauthorized

You are not logged in

Delete All Messages - Admin Only

Delete - http://localhost:8080/chat

Description

If you are signed in as the admin, you will be able to delete all messages. Normally, I would never give the admin credentials on a requirements document, but since this is a graded assignment where testing is required, then refer to the red text in the login endpoint [first endpoint in req. doc).

Responses

201 Created
Created

```
{
   "code": 201,
  "response": "All Messages Deleted"
}
```

401 Unauthorized
You are not logged in as admin

401 Unauthorized
You are not logged in

Add A Message to the Chat

PUT - http://localhost:8080/chat/message/{Lorem Ipsum}

Description

If you are signed in, this request will allow you to write a message. The message will automatically encode the time of the message as well as your account name with it.

Responses

201 Created
Created

```
{
   "code": 201,
  "response": “Message added to chat”:”true”
}
```

401 Unauthorized
You are not logged in

404 Not Found
Check URL parameters

Edit A Previous Message

Post -
http://localhost:8080/chat/time/{foobar}/newMessage/{Lorem Ipsum}

Description

If you are signed in and you have the time of a previous message (in database), this request will allow you to edit said message provided it is your own message.

Responses

201 Created

Created

```
{
   "code": 201,
  "response": “Message at time ‘foobar’ changed
to message ‘Lorem Ipsum’”
}
```

401 Unauthorized

Cannot edit a message you did not create

401 Unauthorized

You are not logged in

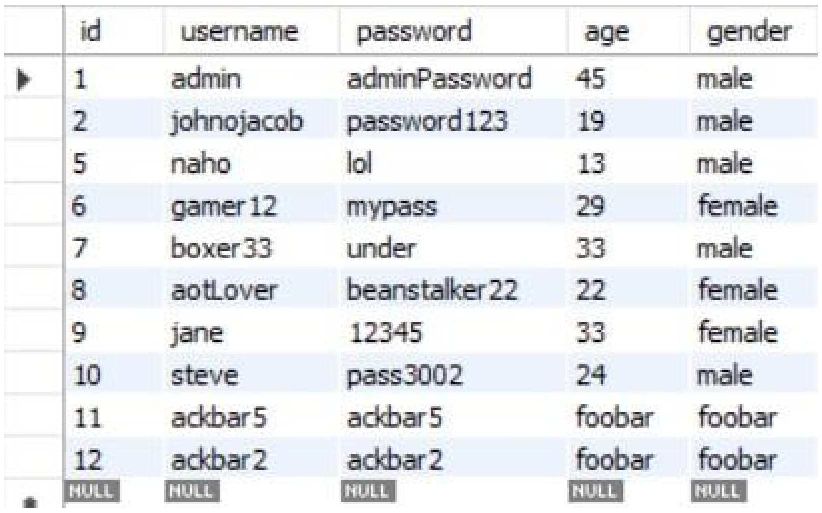
404 Not Found

Check that time entered is valid

<u>MySQL Explanation:</u>

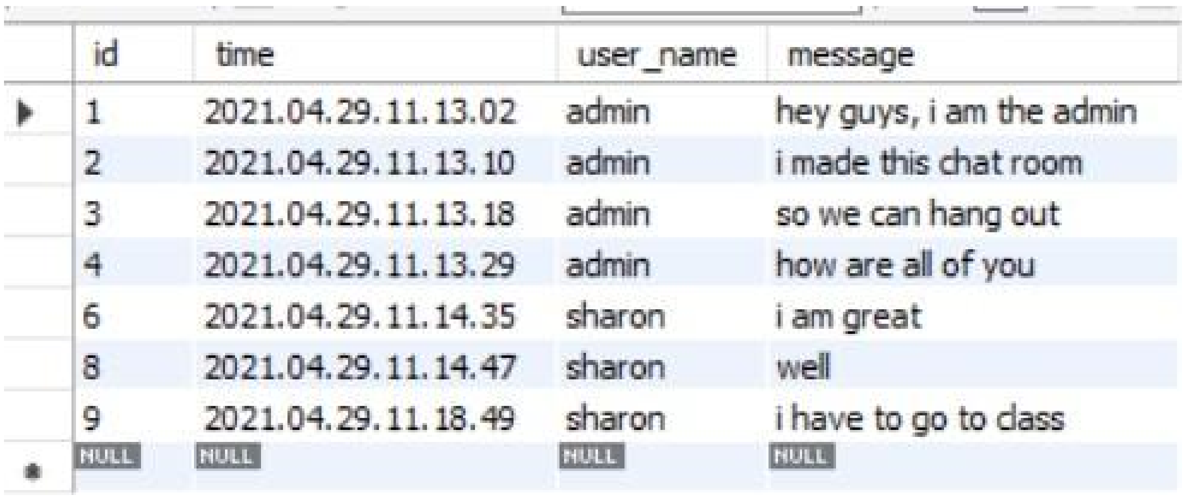
The MySQL commands in the model classes accesses two specific MySQL data dumps with varying fields:

<u>Accounts (example):</u>



id	username	password	age	gender
1	admin	adminPassword	45	male
2	johnojacob	password123	19	male
5	naho	lol	13	male
6	gamer12	mypass	29	female
7	boxer33	under	33	male
8	aotLover	beanstalker22	22	female
9	jane	12345	33	female
10	steve	pass3002	24	male
11	ackbar5	ackbar5	foobar	foobar
12	ackbar2	ackbar2	foobar	foobar
NULL	NULL	NULL	NULL	NULL

<u>Chat Log (example):</u>



id	time	user_name	message
1	2021.04.29.11.13.02	admin	hey guys, i am the admin
2	2021.04.29.11.13.10	admin	i made this chat room
3	2021.04.29.11.13.18	admin	so we can hang out
4	2021.04.29.11.13.29	admin	how are all of you
6	2021.04.29.11.14.35	sharon	i am great
8	2021.04.29.11.14.47	sharon	well
9	2021.04.29.11.18.49	sharon	i have to go to class
NULL	NULL	NULL	NULL

Authentication:

Upon logging in, a programmatically based token is sent on the server side consisting of the logged in user's username. This token will remain until connection is lost (server is taking down for example) or the user utilizes the log out endpoint. The token is stored in a session for later use.

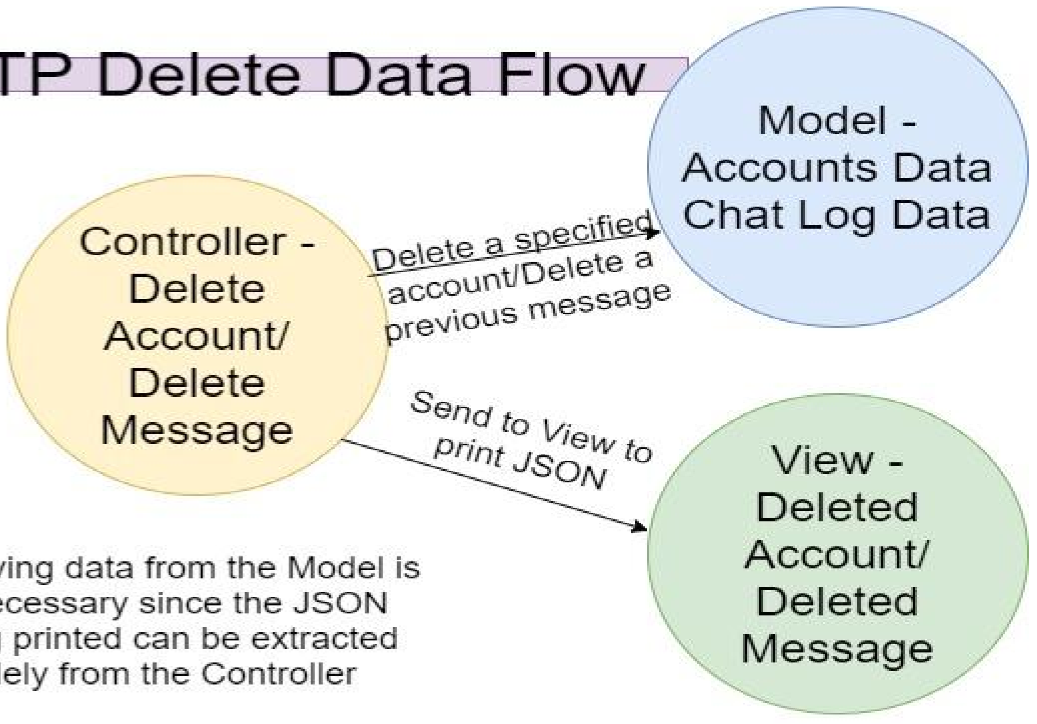
Diagrammatic Logic:

The web application is implemented with an MVC. A controller sends a request to the Model class(es) and retrieves data (if necessary) back to the controller. The controller then sends said date (from model or itself) to the View in json format.

Data Flow:

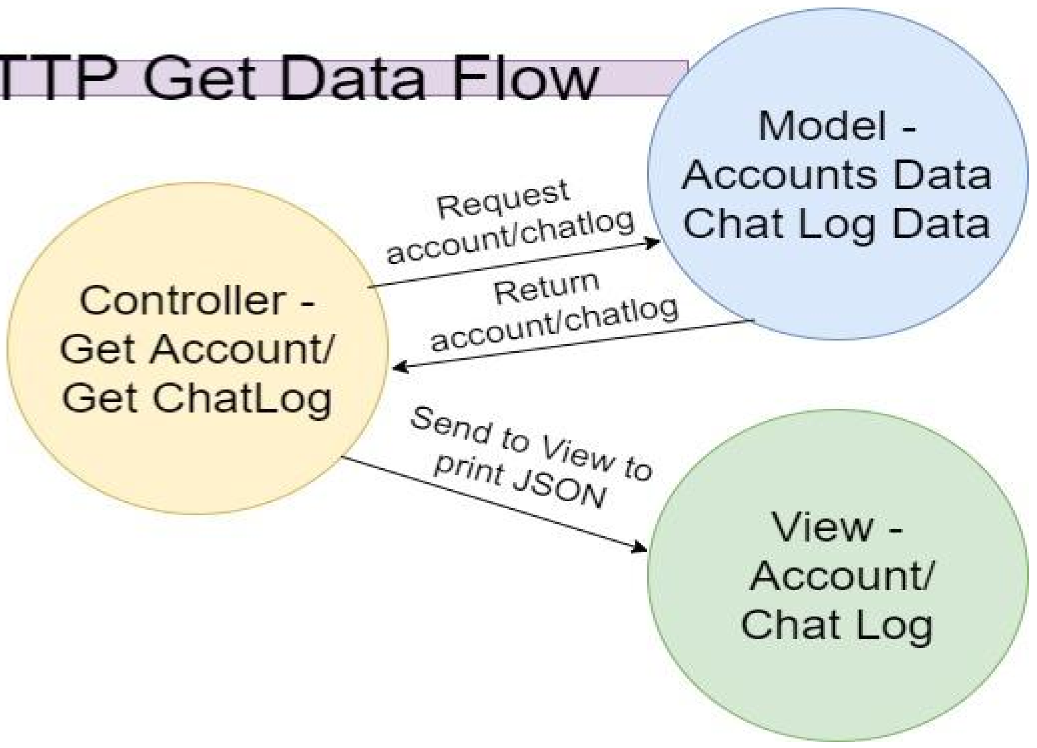
Data flow varies based on each HTTP request. Each HTTP request interacts with the Model (even if not retrieving from) and utilizes MySQL commands to alter MySQL databases (accounts and chat log). There are attached diagrams below.

HTTP Delete Data Flow

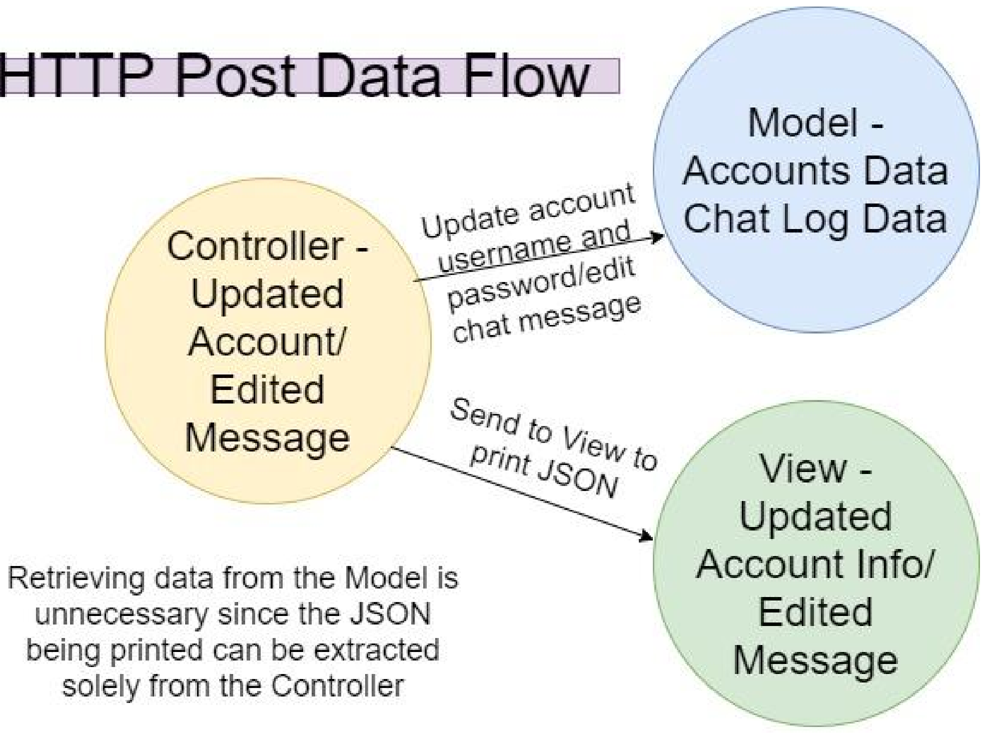


Retrieving data from the Model is unnecessary since the JSON being printed can be extracted solely from the Controller

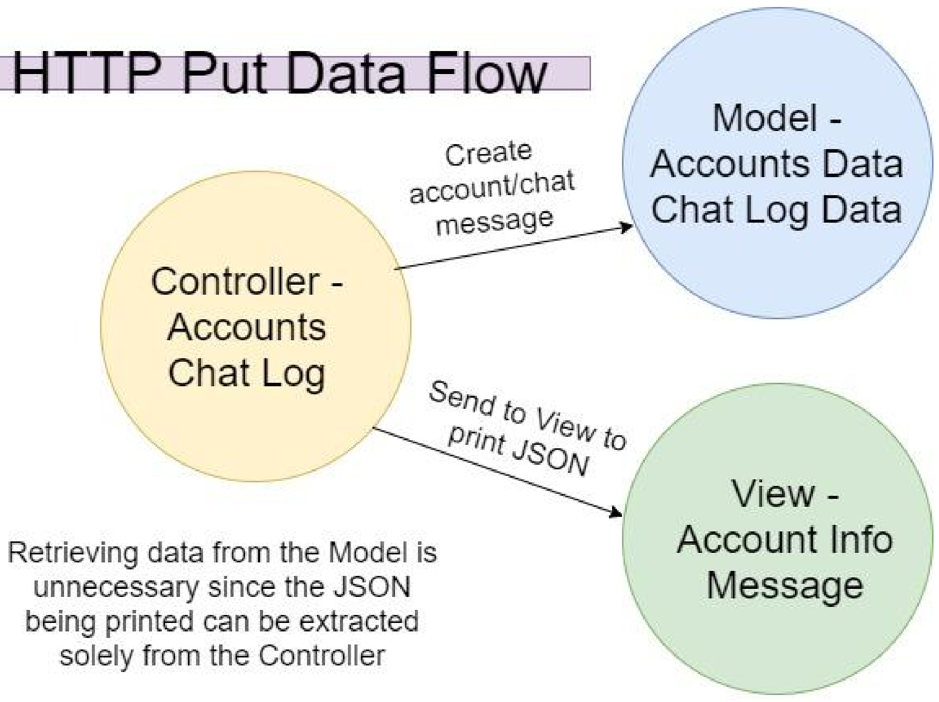
HTTP Get Data Flow



HTTP Post Data Flow


Model - Accounts Data Chat Log Data
Controller - Updated Account/ Edited Message
Update account username and password/edit chat message
Send to View to print JSON
View - Updated Account Info/ Edited Message
Retrieving data from the Model is unnecessary since the JSON being printed can be extracted solely from the Controller

HTTP Put Data Flow


Model - Accounts Data Chat Log Data
Create account/chat message
Controller - Accounts Chat Log
Send to View to print JSON
View - Account Info Message
Retrieving data from the Model is unnecessary since the JSON being printed can be extracted solely from the Controller

Additional Notes:

To successfully connect with the database, the parameters (‘user’ and ‘password’) inside of “lol.yml” below must be changed to account for the client’s local machine’s MySQL root username and root password.

```
user: x

password: x
```