Web Studio: Week 7 Turn-in #6

Name: Ivy Chen
UNI: ic2389
Due: Friday 10/19 at 2PM
COMS6998 Adv. Web Studio

Original Goals

High Level Goal

1. Implement the real-time synchronous group chat aspect of Twitter

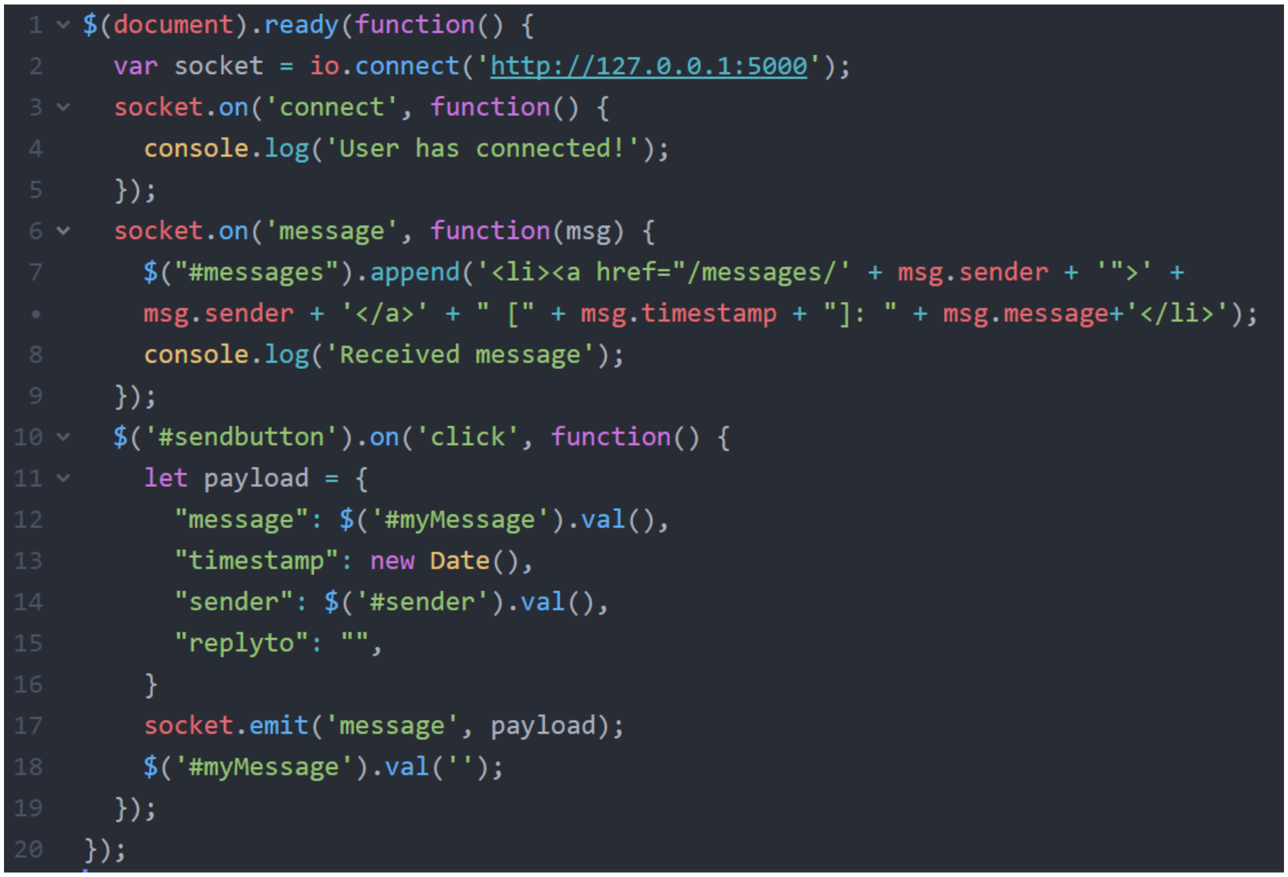
Low Level Goal

1. Add user accounts
2. Set up DB to store chat history
3. Create DB schemas to store sender name, timestamp, message and who it replies to (if reply)
4. Update the JS on message received to include sender name and timestamp on chats that appear in real-time
5. Create homepage of all messages
6. Create pages for individual users messages
7. Update message handler to store extra information about a message
8. Allow users to reply to messages
9. Stretch goal: allow for "chat" messages between two users

Lessons Learned Through Iteration

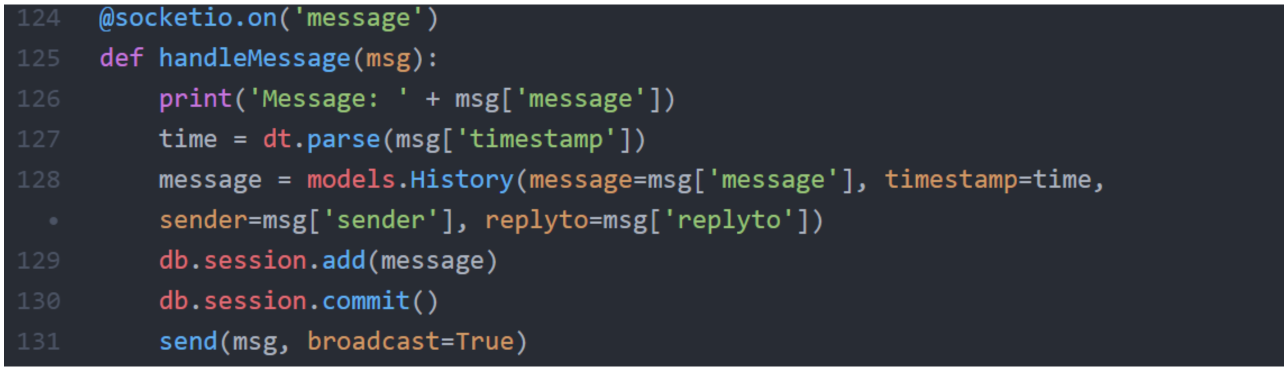
Report on 3 of the things that you learned that caused you to iterate. Consider the following format: My plan was to __x___. But I ran into problem __y___. And I solved it by doing __z___.
Where z = "I added a new subgoal", "I changed my high level goal", "I removed a subgoal" Each of the three things should take 1-2 paragraphs to describe. Images are encouraged.

1. My plan was to send message, sender, replyto and a timestamp to the database in the socket.io message handler. But I ran into issues getting user information from flask-login's `current_user` in my message handler. It turns out that you need to do additional setup to log sessions and pass them to socket.io from flask-login because they don't share sessions. I tried looking through the documentation and saw similar questions online, but ultimately couldn't figure out how to get the user session in my message handler.
   Since my main goal was to pass sender and timestamp, I decided to create an input field on the messages page that was automatically populated with the sender username. Then in my JS handler for sending messages, I retrieved the value from the input field and sent it to socket.io to process.



```
$(document).ready(function() {
  var socket = io.connect('http://127.0.0.1:5000');
  socket.on('connect', function() {
    console.log('User has connected!');
  });
  socket.on('message', function(msg) {
    $("#messages").append('<li><a href="/messages/' + msg.sender + '">' +
    msg.sender + '</a>' + " [" + msg.timestamp + "]: " + msg.message+'</li>');
    console.log('Received message');
  });
  $('#sendbutton').on('click', function() {
    let payload = {
      "message": $('#myMessage').val(),
      "timestamp": new Date(),
      "sender": $('#sender').val(),
      "replyto": "",
    }
    socket.emit('message', payload);
    $('#myMessage').val('');
  });
});
```

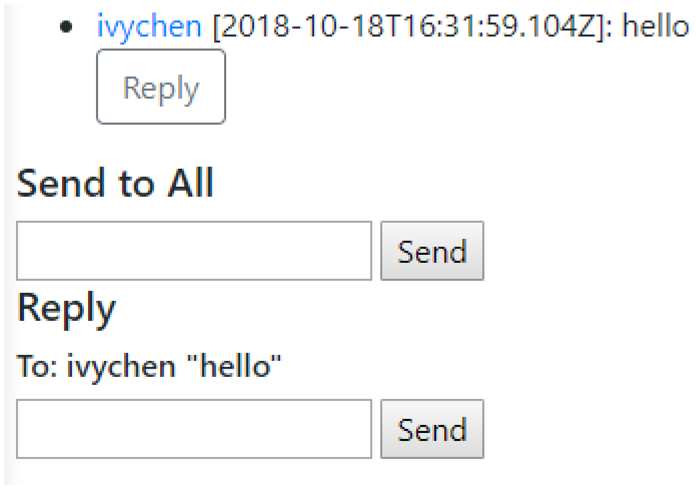
main.js for retrieving metadata about message



```python
@socketio.on('message')
def handleMessage(msg):
    print('Message: ' + msg['message'])
    time = dt.parse(msg['timestamp'])
    message = models.History(message=msg['message'], timestamp=time, sender=msg['sender'], replyto=msg['replyto'])
    db.session.add(message)
    db.session.commit()
    send(msg, broadcast=True)
```

socket.io handler for message

2. I wanted to create a reply widget similar to how Twitter handles their replies. Ie. a thread under each main post. However, I had trouble just getting the reply to work correctly ie. identifying which message I was replying to, and figuring out how to send information.
   Thus for this week, I made a super simple reply mechanism where there’s a single reply input field at the bottom of the page, each message has a “Reply” button, and clicking on it will prepopulate the input field with metadata. Then a user can type a message reply and click “Reply”. Unfortunately, you can’t tell which message a reply is for in the current app because I simply append the message to the list of messages. Ideally I want this to be cleaner and have a sort of “thread” format.



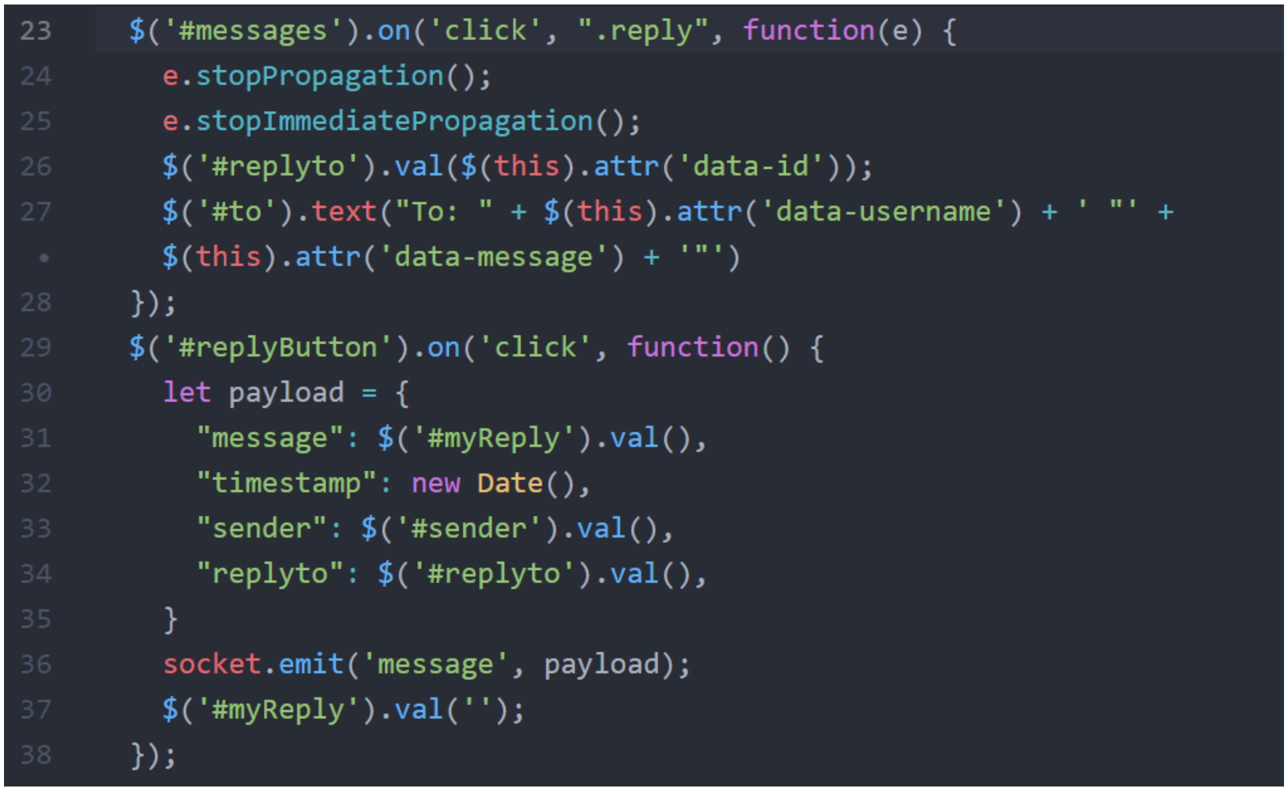
- ivychen [2018-10-18T16:31:59.104Z]: hello

Send to All

Reply

To: ivychen "hello"

Main page input for replying to messages



```
  $('#messages').on('click', ".reply", function(e) {
    e.stopPropagation();
    e.stopImmediatePropagation();
    $('#replyto').val($(this).attr('data-id'));
    $('#to').text("To: " + $(this).attr('data-username') + ' "' +
    $(this).attr('data-message') + '"')
  });
  $('#replyButton').on('click', function() {
    let payload = {
      "message": $('#myReply').val(),
      "timestamp": new Date(),
      "sender": $('#sender').val(),
      "replyto": $('#replyto').val(),
    }
    socket.emit('message', payload);
    $('#myReply').val('');
  });
```

Main.js handler

3. One of my goals was to enable chatting between two users. So a user can click on another user and it would take them to a chat page for the two of them. However, I wasn't sure where these messages would fit in since I currently don't have any sort of delineation between "public" and 'private" messages. Currently, all messages and replies are public to everyone, but for chat, I'd likely want to have some notion of message privacy. I thought it would be confusing that clicking on another person's username takes them to a view of their public messages instead of chat.

   I removed the sub-goal for now until I build out the public message feature further. Then later on it would be cool to introduce direct messages.

<u>Goal Progress</u>

Status Summary

✓ = DONE
🔜 = PARTIAL/DEFERRED

✕ = NOT DONE (REMOVED)

High Level Goal	**Status**
1. Implement the real-time synchronous group chat aspect of Twitter	✓
Low Level Goals	**Status**
1. Add user accounts	✓
2. Create DB schemas to store sender name, timestamp, message and who it replies to (if reply)	✓
3. Update the JS on message received to include sender name and timestamp on chats that appear in real-time	✓
4. Create homepage of all messages	✓
5. Create pages for individual users messages	✓
6. Update message handler to store extra information about a message	✓
7. Allow users to reply to messages	✓

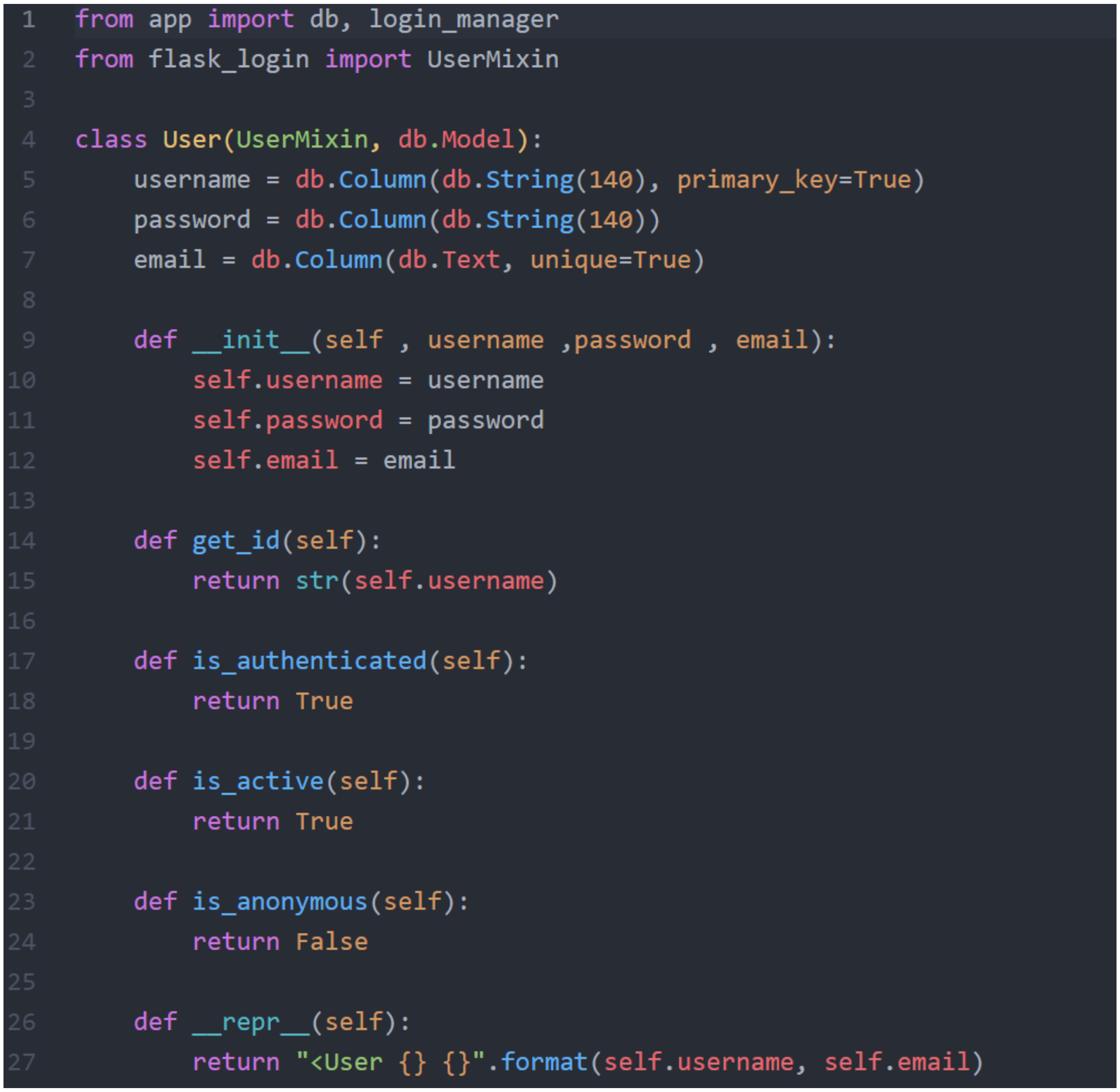
High Level Goal

1. Implement the real-time synchronous group chat aspect of Twitter

Used the Flask socket.io tutorials to help me build the handlers and understand how to send messages, and add chat history. I set up my app so that a user must sign in first before being able to send a message ie. they can't be anonymous.

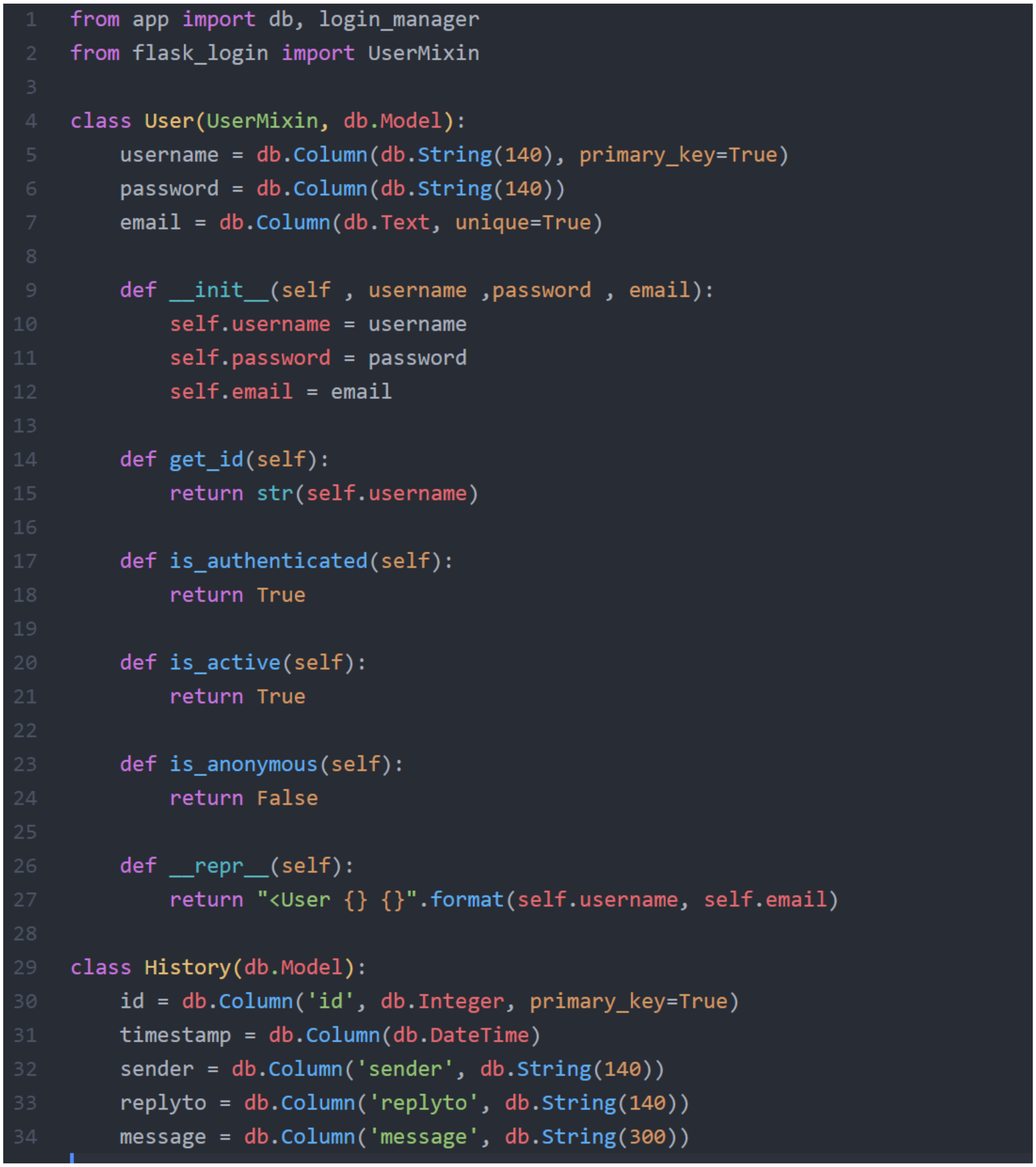
Low Level Goal

1. Add user accounts
   Added user accounts by re-using code from the IMDB project for login and register pages. I enforced that users must sign-in before being able to send any messages, but they can see messages even if they haven't signed in.



```python
from app import db, login_manager
from flask_login import UserMixin

class User(UserMixin, db.Model):
    username = db.Column(db.String(140), primary_key=True)
    password = db.Column(db.String(140))
    email = db.Column(db.Text, unique=True)

    def __init__(self , username ,password , email):
        self.username = username
        self.password = password
        self.email = email

    def get_id(self):
        return str(self.username)

    def is_authenticated(self):
        return True

    def is_active(self):
        return True

    def is_anonymous(self):
        return False

    def __repr__(self):
        return "<User {} {}".format(self.username, self.email)
```

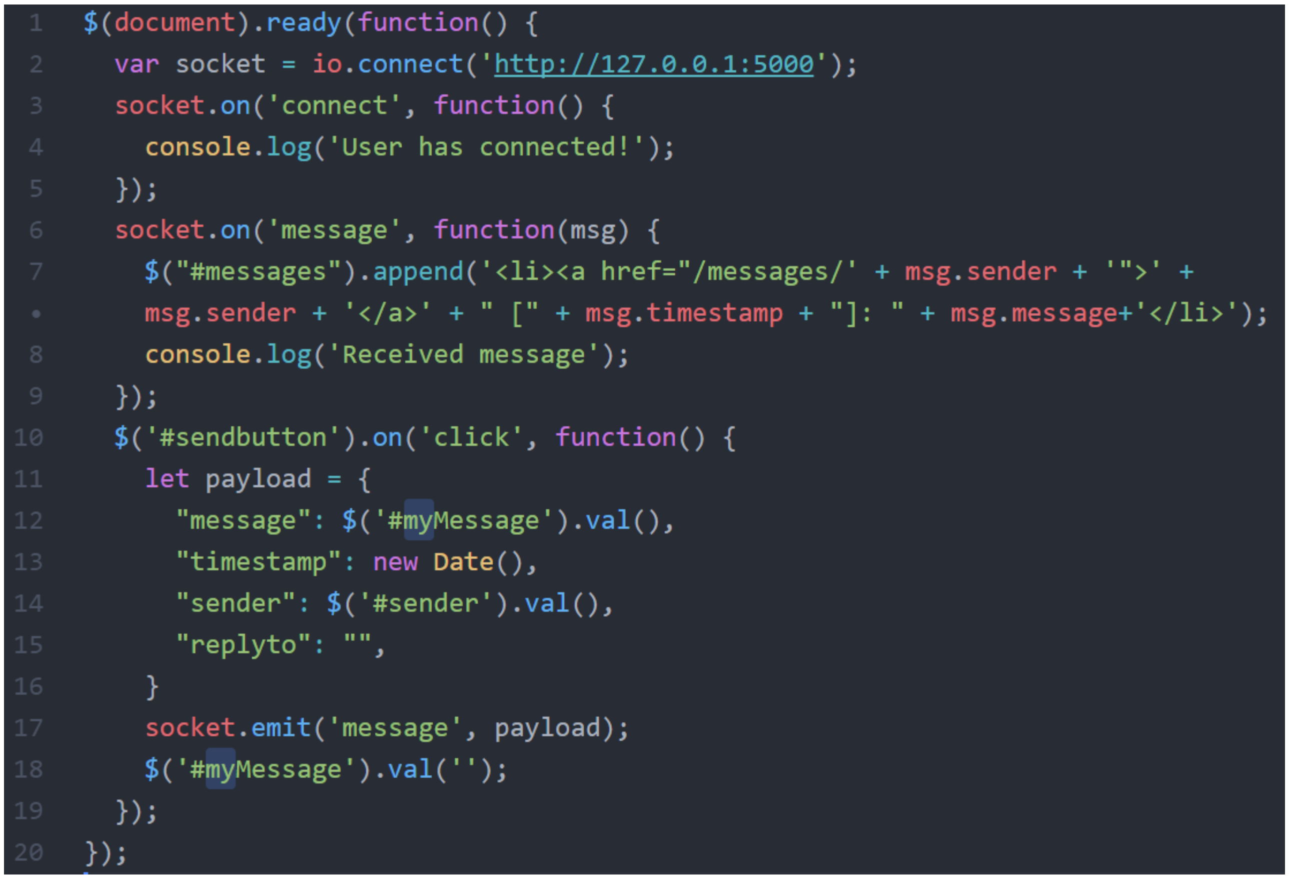
2. Create DB schemas to store sender name, timestamp, message and replyto (if reply)
   I created a SQLite database and schemas to store chat history. On the main page, all records from the DB are retrieved. I decided to augment the schema to store sender name, timestamp and replyto in addition to the message content.



```python
from app import db, login_manager
from flask_login import UserMixin

class User(UserMixin, db.Model):
    username = db.Column(db.String(140), primary_key=True)
    password = db.Column(db.String(140))
    email = db.Column(db.Text, unique=True)

    def __init__(self , username ,password , email):
        self.username = username
        self.password = password
        self.email = email

    def get_id(self):
        return str(self.username)

    def is_authenticated(self):
        return True

    def is_active(self):
        return True

    def is_anonymous(self):
        return False

    def __repr__(self):
        return "<User {} {}".format(self.username, self.email)

class History(db.Model):
    id = db.Column('id', db.Integer, primary_key=True)
    timestamp = db.Column(db.DateTime)
    sender = db.Column('sender', db.String(140))
    replyto = db.Column('replyto', db.String(140))
    message = db.Column('message', db.String(300))
```

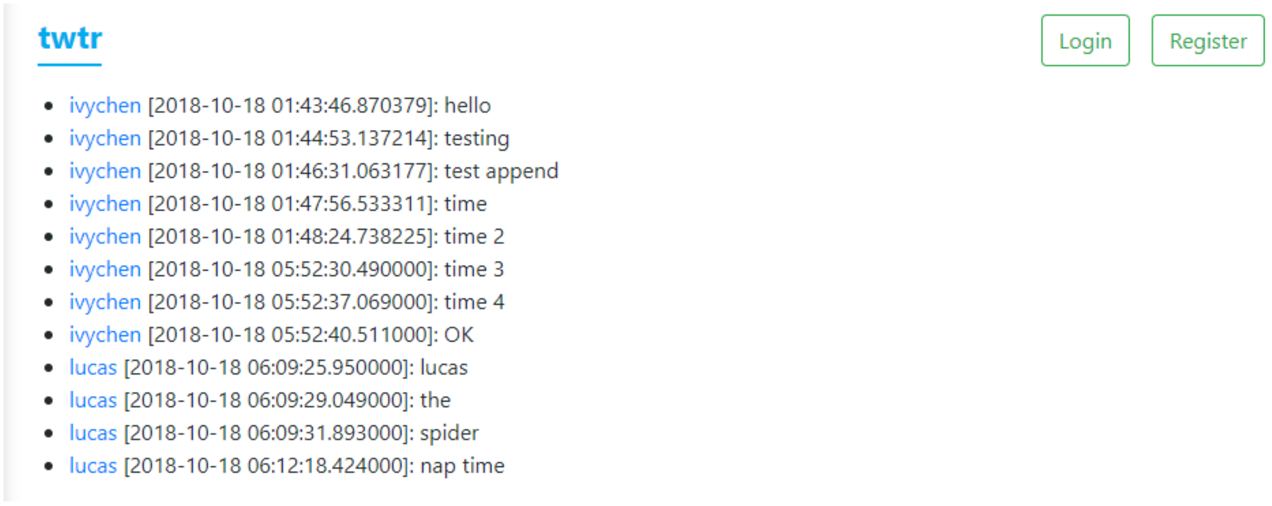
models.py

3. Update the JS on message received to include sender name and timestamp on chats that appear in real-time
   Since I needed to record the sender name, I updated the message payload send over the socket.io connection to include metadata such as sender name in addition to timestamp and replyto name.



```
$(document).ready(function() {
  var socket = io.connect('http://127.0.0.1:5000');
  socket.on('connect', function() {
    console.log('User has connected!');
  });
  socket.on('message', function(msg) {
    $("#messages").append('<li><a href="/messages/' + msg.sender + '">' +
    msg.sender + '</a>' + " [" + msg.timestamp + "]: " + msg.message+'</li>');
    console.log('Received message');
  });
  $('#sendbutton').on('click', function() {
    let payload = {
      "message": $('#myMessage').val(),
      "timestamp": new Date(),
      "sender": $('#sender').val(),
      "replyto": "",
    }
    socket.emit('message', payload);
    $('#myMessage').val('');
  });
});
```

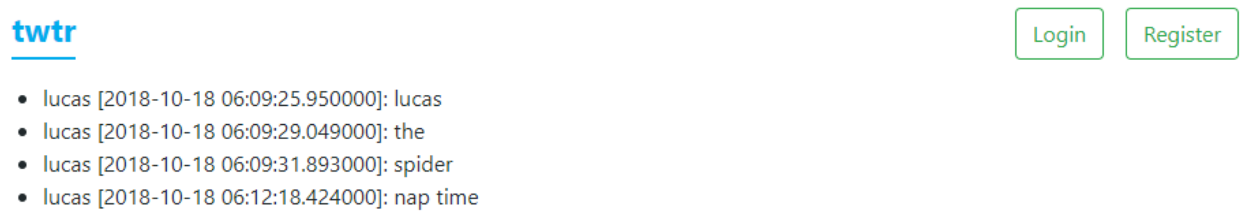
4. Create homepage of all messages
   I created the homepage with the help of the flask-socketio tutorial to display all messages. However a user must login before they can send a message. Each message includes the sender's name and timestamp.



- ivychen [2018-10-18 01:43:46.870379]: hello
- ivychen [2018-10-18 01:44:53.137214]: testing
- ivychen [2018-10-18 01:46:31.063177]: test append
- ivychen [2018-10-18 01:47:56.533311]: time
- ivychen [2018-10-18 01:48:24.738225]: time 2
- ivychen [2018-10-18 05:52:30.490000]: time 3
- ivychen [2018-10-18 05:52:37.069000]: time 4
- ivychen [2018-10-18 05:52:40.511000]: OK
- lucas [2018-10-18 06:09:25.950000]: lucas
- lucas [2018-10-18 06:09:29.049000]: the
- lucas [2018-10-18 06:09:31.893000]: spider
- lucas [2018-10-18 06:12:18.424000]: nap time

5. Create pages for individual users messages

By clicking on a user's name from the home page, you can find all the messages for a given user. Anyone can access this information.

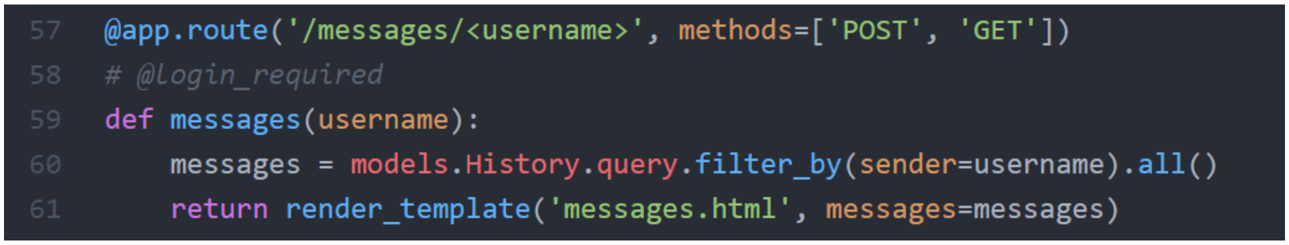


twtr

Login Register

- lucas [2018-10-18 06:09:25.950000]: lucas
- lucas [2018-10-18 06:09:29.049000]: the
- lucas [2018-10-18 06:09:31.893000]: spider
- lucas [2018-10-18 06:12:18.424000]: nap time

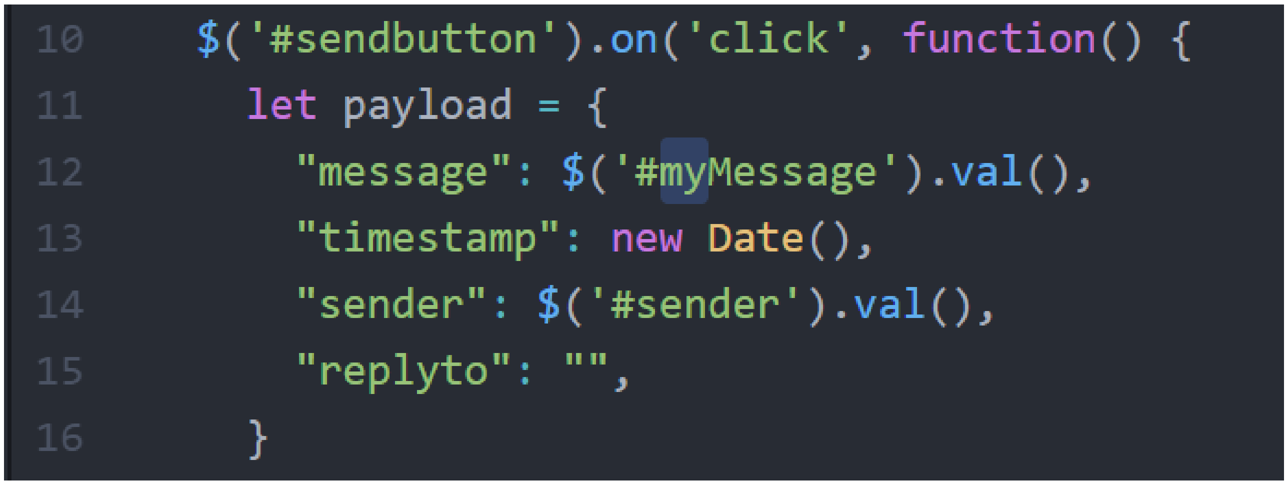
Lucas's messages



```
@app.route('/messages/<username>', methods=['POST', 'GET'])
# @login_required
def messages(username):
    messages = models.History.query.filter_by(sender=username).all()
    return render_template('messages.html', messages=messages)
```

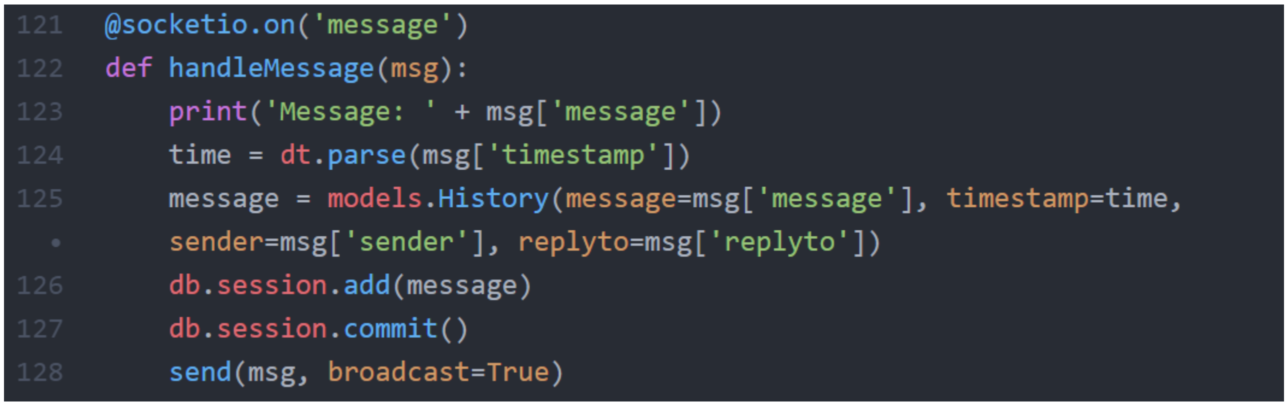
Code for user message page

6. Update message handler to store extra information about a message
   I needed to store metadata about a message so I had to update my socket.io message handler as well as the JS to pass data through the connection.



```
    $('#sendbutton').on('click', function() {
      let payload = {
        "message": $('#myMessage').val(),
        "timestamp": new Date(),
        "sender": $('#sender').val(),
        "replyto": "",
      }
```

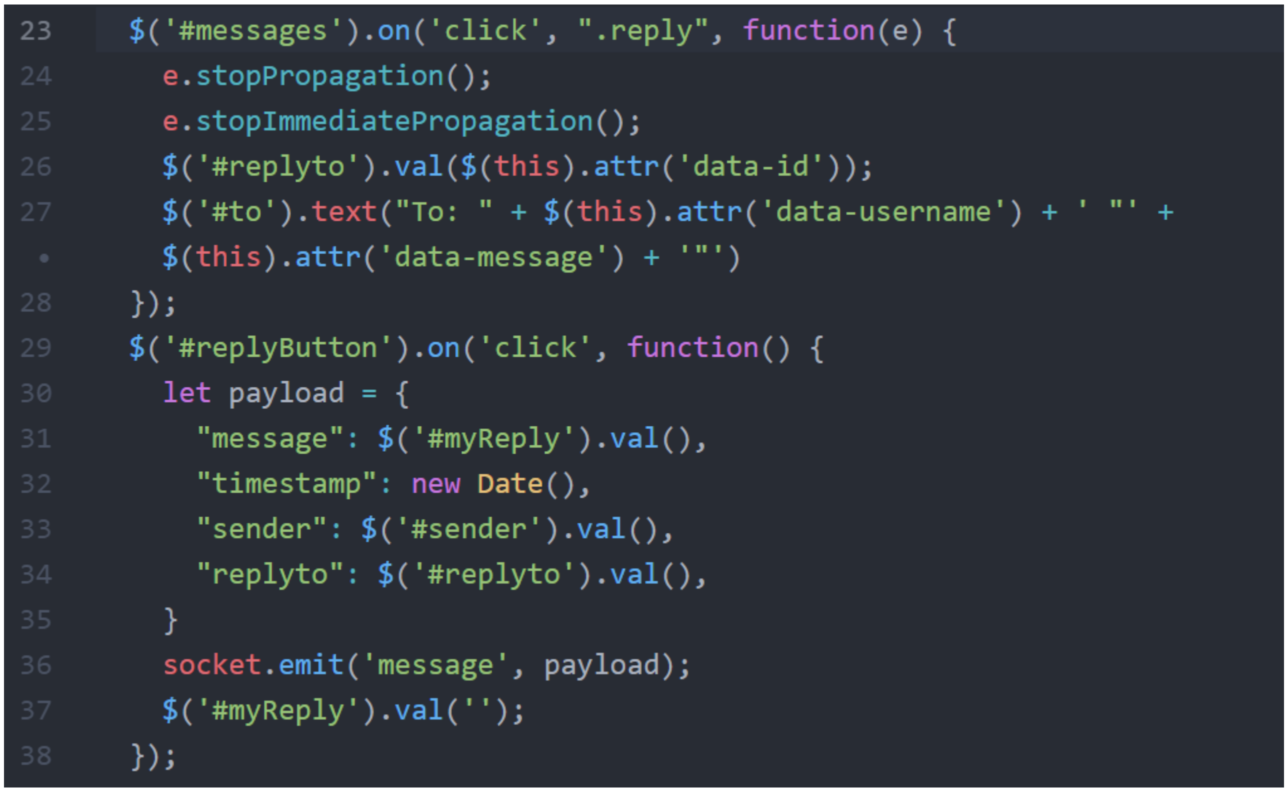
Main.js handler



```python
@socketio.on('message')
def handleMessage(msg):
    print('Message: ' + msg['message'])
    time = dt.parse(msg['timestamp'])
    message = models.History(message=msg['message'], timestamp=time, sender=msg['sender'], replyto=msg['replyto'])
    db.session.add(message)
    db.session.commit()
    send(msg, broadcast=True)
```

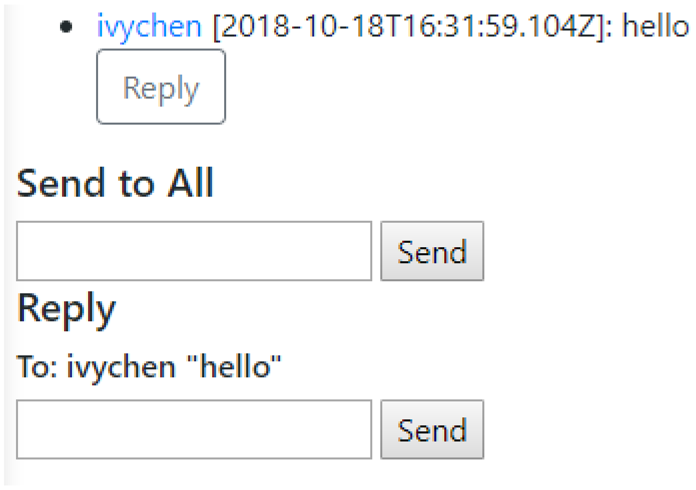
Message handler

7. Allow users to reply to messages
   I created another input field to allow users to reply to messages. Next to each message is a reply button. When a user clicks, I populate the reply input fields with the appropriate metadata. Users can reply to their own messages.



```javascript
  $('#messages').on('click', ".reply", function(e) {
    e.stopPropagation();
    e.stopImmediatePropagation();
    $('#replyto').val($(this).attr('data-id'));
    $('#to').text("To: " + $(this).attr('data-username') + ' "' + $(this).attr('data-message') + '"')
  });
  $('#replyButton').on('click', function() {
    let payload = {
      "message": $('#myReply').val(),
      "timestamp": new Date(),
      "sender": $('#sender').val(),
      "replyto": $('#replyto').val(),
    }
    socket.emit('message', payload);
    $('#myReply').val('');
  });
```

Main.js handler



- ivychen [2018-10-18T16:31:59.104Z]: hello

Send to All

Reply

To: ivychen "hello"

Main page input for replying to messages