

COS60016: Programming for Development

Assignment 2: Build a chatbot

Student: Alexandra Bain

Student Email: 103083826

Student ID: 103083826@student.swin.edu.au

## Chapters

<i>1. Analysis of chatbot design</i>	
<b>Chatbot Creation</b>	3
<b>Chatterbot and Flask App Configuration</b>	3
<b>Chatbot Training</b>	4
<i>2. Python code to build the chatbot</i>	
<b>Building the Bot App</b>	9
<b>Pulling Data from OpenWeatherMaps API</b>	13
<b>Pulling Data through CSV and Pandas</b>	15
<b>Wrangling Data Using DB Browser and SQLite</b>	17
<b>Calling Data from Data Class</b>	20
<b>Chatbot Javascript</b>	21
<i>3. Reflection</i>	28

## 1. Analysis of chatbot design

### Chatbot Creation

I will be building a chatbot able to integrate with the original weather application I built for the “Go Travel!” website. I set up a Flask app to pull data, with a route for displaying chatbot responses to train the chatbot.

### Chatterbot and Flask App Configuration

To start with building the weather chatbot for the blogger’s website, I created a fresh Python project after downloading and downgrading to python 3.7.9 to download the Chatterbot, Flask and SQLAlchemy libraries. I experienced a few compatibility issues with libraries and striking the right balance/ researching what was needed. For example, I had some issues with Spacy, which downloaded automatically with ChatterBot and was affecting other dependencies on my app. I managed to work around this by downloading ChatterBot and re-installing Stacy and then re-wrote the tagging.py non-project file (that kept being flagged as an error) without any reference to Spacy.

I initialise the app with Socket.io library, a JS library that enables real-time, bidirectional communication between web clients and servers, to keep the chatbot’s weather information responses up-to-date.

I initialised my Chatterbot chatbot and named it AllieChat:

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top indicates the project is named 'pythonProject3' and is currently on the 'master' branch. The left sidebar displays the project structure, including a 'chat\_files' folder containing 'index.html', 'app.py', and 'allieChat.js'; a 'static' folder with 'css' and 'js' subfolders; and a 'templates' folder with 'index.html', 'explore\_data.html', and 'newest\_chat.db'. The main code editor window shows the 'app.py' file with the following code:

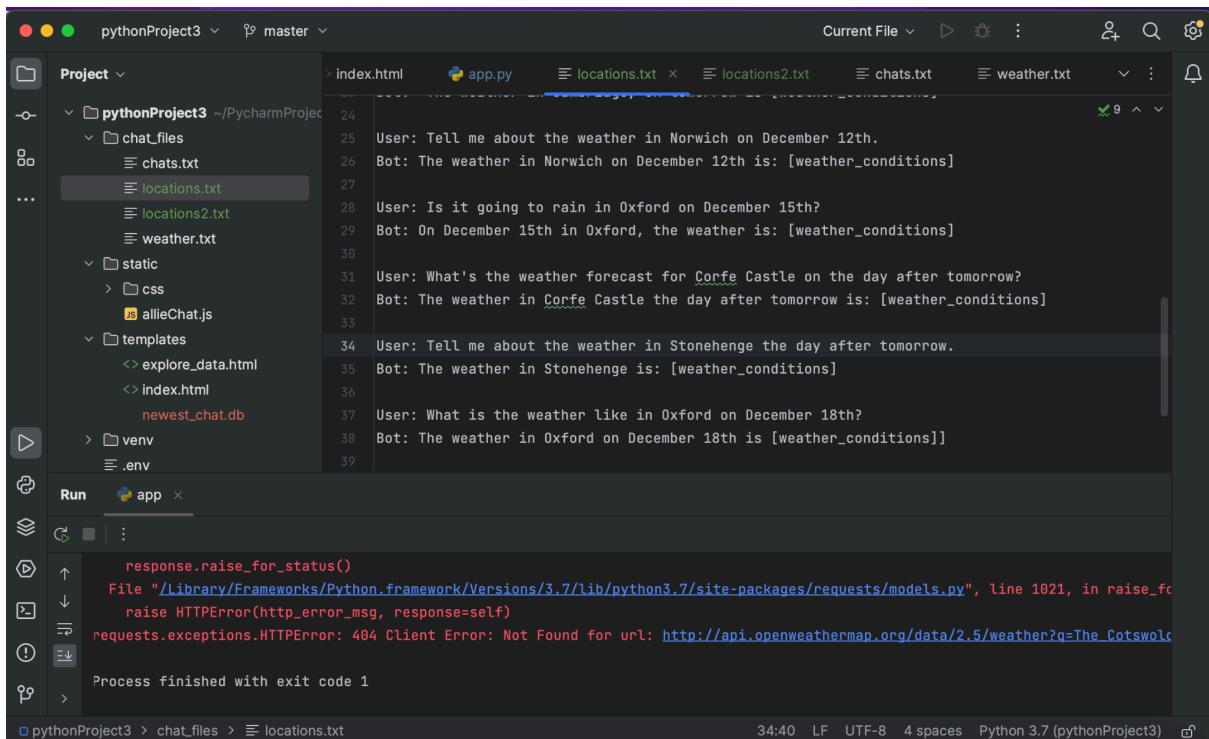
```
# creating my chatbot instance:  
my_bot = ChatBot(  
    'Allie',  
    storage_adapter='chatterbot.storage.SQLStorageAdapter',  
    database_uri='sqlite:///newest_chat.db',  
    logic_adapters=[  
        'chatterbot.logic.MathematicalEvaluation',  
        'chatterbot.logic.BestMatch'  
    ]  
)  
  
#training the bot, here with Data class:  
list_trainer = ListTrainer(my_bot)  
try:  
    for item in Data.query.all():  
        fetch_openweathermap_data_for_mult... > for city in cities > try
```

I added an index.html template page in the ‘templates’ directory, and a static directory with an “allieChat.js” Javascript, which implemented chat functionality.

## Chatbot Training

I trained the chatbot with conversation and general weather information to prepare for importing database info.

I included examples of user queries and correct responses to improve the chatbot's ability to handle a wider range of inputs. I trained with user queries that the chatbot may not handle well and got as specific as possible. I also linked it to the function `weather_queries` in `app.py`:



```
User: Tell me about the weather in Norwich on December 12th.  
Bot: The weather in Norwich on December 12th is: [weather_conditions]  
  
User: Is it going to rain in Oxford on December 15th?  
Bot: On December 15th in Oxford, the weather is: [weather_conditions]  
  
User: What's the weather forecast for Corfe Castle on the day after tomorrow?  
Bot: The weather in Corfe Castle the day after tomorrow is: [weather_conditions]  
  
User: Tell me about the weather in Stonehenge the day after tomorrow.  
Bot: The weather in Stonehenge is: [weather_conditions]  
  
User: What is the weather like in Oxford on December 18th?  
Bot: The weather in Oxford on December 18th is [weather_conditions]  
  
Process finished with exit code 1
```

I also had to account for the fact that my API in `app.py` was *not* pulling information directly from The Cotswolds, but instead, Gloucestershire and instead of Watergate Bay, Newquay. Therefore, I was sure to include these locations in the text chats:

```

User: How hot is it tomorrow?
Bot: Please be more specific.

User: What is the weather in The Cotswolds like on Dec 11?
Bot: The weather in The Cotswolds, Gloucestershire, UK is [weather_conditions]

User: Is it going to rain tomorrow in Norwich?
Bot: the weathe tomorrow in Norwich is [weather_conditions]

User: Is it going to rain tomorrow in Birmingham
Bot: The weather tomorrow in Birmingham tomorrow is: [weather_conditions]

User: What is the weather in Watergate Bay tomorrow?
Bot: The Weather in Newquay is [weather_conditions]

User: What is the weather like in the Cotswolds next week?
Bot: the weather next week in The Cotswolds, Gloucestershire is [weather_conditions]

User: What is the weather in Corfe Castle, UK tomorrow?
Bot: The weather in Corfe Castle, UK is [weather_conditions]

User: What is the weather like in Norwich in two days?
Bot: The weather in Norwich in the next two days is [weather_conditions]

User: What is the weather like in Cambridge, UK tomorrow?
Bot: The weather in Cambridge, UK tomorrow is [weather_conditions]

```

I implemented A/B Testing:

- I built multiple chatbot versions across two different app structures, using Python Flask. Experimenting with these different versions and the varying output of the chatbots, I was able to see which performed better, and;
- I tested variations in language and response structure.

```

# running the app:
if __name__ == '__main__':
    try:
        # Print the database file path for debugging purposes
        print(f"Database file path: {app.config['SQLALCHEMY_DATABASE_URI']}")

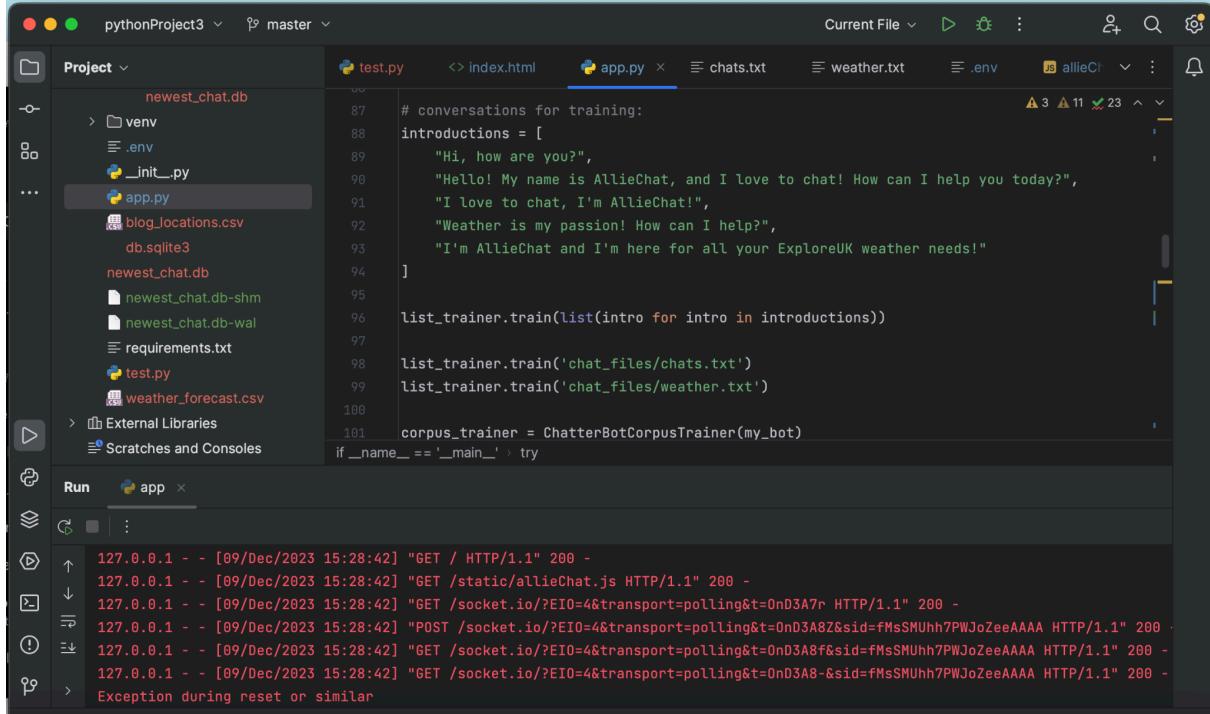
        # Creating the database tables- this was a huge step I had initially missed!
        db.create_all()

        # Run the Flask app with SocketIO
        socketio.run(app, debug=True, allow_unsafe_werkzeug=True, use_reloader=False)

    except Exception as e:
        print(f"Error creating database or running app: {e}")
        import traceback

```

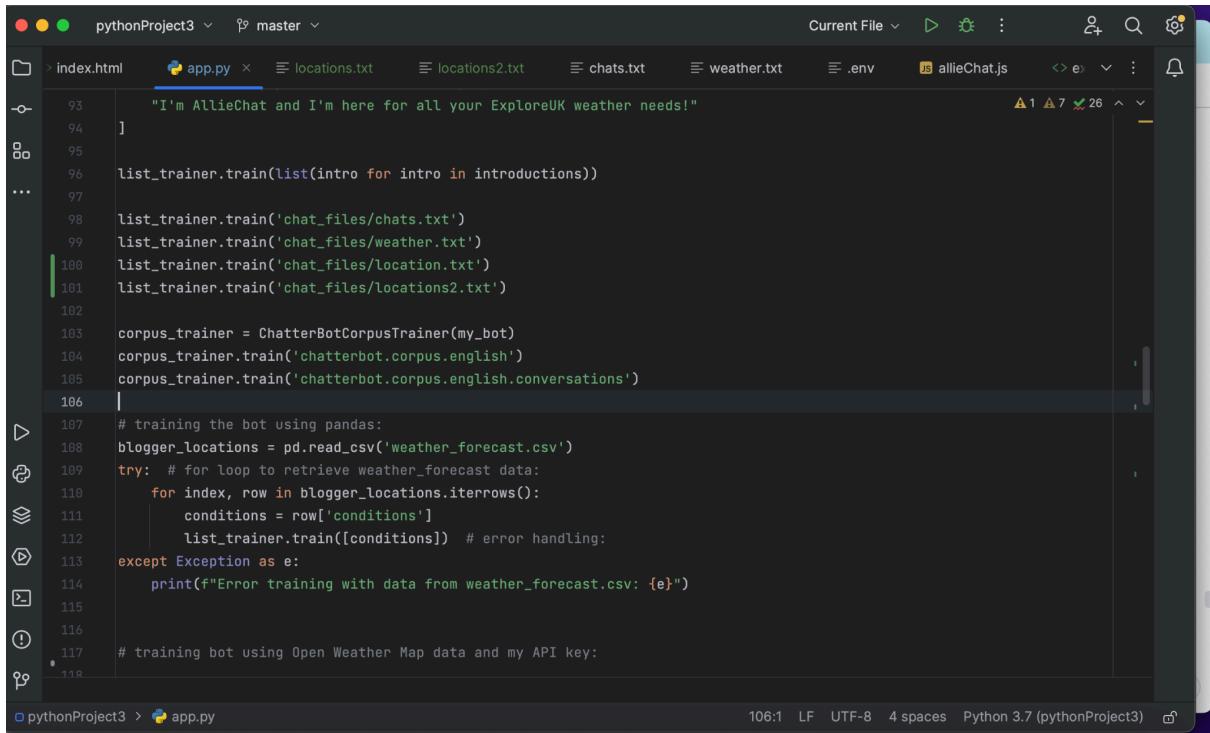
I added in ChatterBot Corpus English and Conversation trainers, and some simple AllieChat greetings, “introductions”, in a list to loop through:



The screenshot shows the VS Code interface with the project 'pythonProject3' open. The 'app.py' file is the active tab, displaying Python code for training a chatbot. The code includes a list of introductory messages and calls to train the list\_trainer and corpus\_trainer. The terminal below shows a log of requests from 127.0.0.1, indicating the bot is running and responding to socket.io connections.

```
# conversations for training:  
introductions = [  
    "Hi, how are you?",  
    "Hello! My name is AllieChat, and I love to chat! How can I help you today?",  
    "I love to chat, I'm AllieChat!",  
    "Weather is my passion! How can I help?",  
    "I'm AllieChat and I'm here for all your ExploreUK weather needs!"  
]  
  
list_trainer.train(list(intro for intro in introductions))  
  
list_trainer.train('chat_files/chats.txt')  
list_trainer.train('chat_files/weather.txt')  
  
corpus_trainer = ChatterBotCorpusTrainer(my_bot)  
if __name__ == '__main__': try:  
  
127.0.0.1 - - [09/Dec/2023 15:28:42] "GET / HTTP/1.1" 200 -  
127.0.0.1 - - [09/Dec/2023 15:28:42] "GET /static/allieChat.js HTTP/1.1" 200 -  
127.0.0.1 - - [09/Dec/2023 15:28:42] "GET /socket.io/?EIO=4&transport=polling&t=0nD3A7r HTTP/1.1" 200 -  
127.0.0.1 - - [09/Dec/2023 15:28:42] "POST /socket.io/?EIO=4&transport=polling&t=0nD3A8Z&sid=fMsSMUhh7PWJoZeeAAAAA HTTP/1.1" 200 -  
127.0.0.1 - - [09/Dec/2023 15:28:42] "GET /socket.io/?EIO=4&transport=polling&t=0nD3A8f&sid=fMsSMUhh7PWJoZeeAAAAA HTTP/1.1" 200 -  
127.0.0.1 - - [09/Dec/2023 15:28:42] "GET /socket.io/?EIO=4&transport=polling&t=0nD3A8-&sid=fMsSMUhh7PWJoZeeAAAAA HTTP/1.1" 200 -  
Exception during reset or similar
```

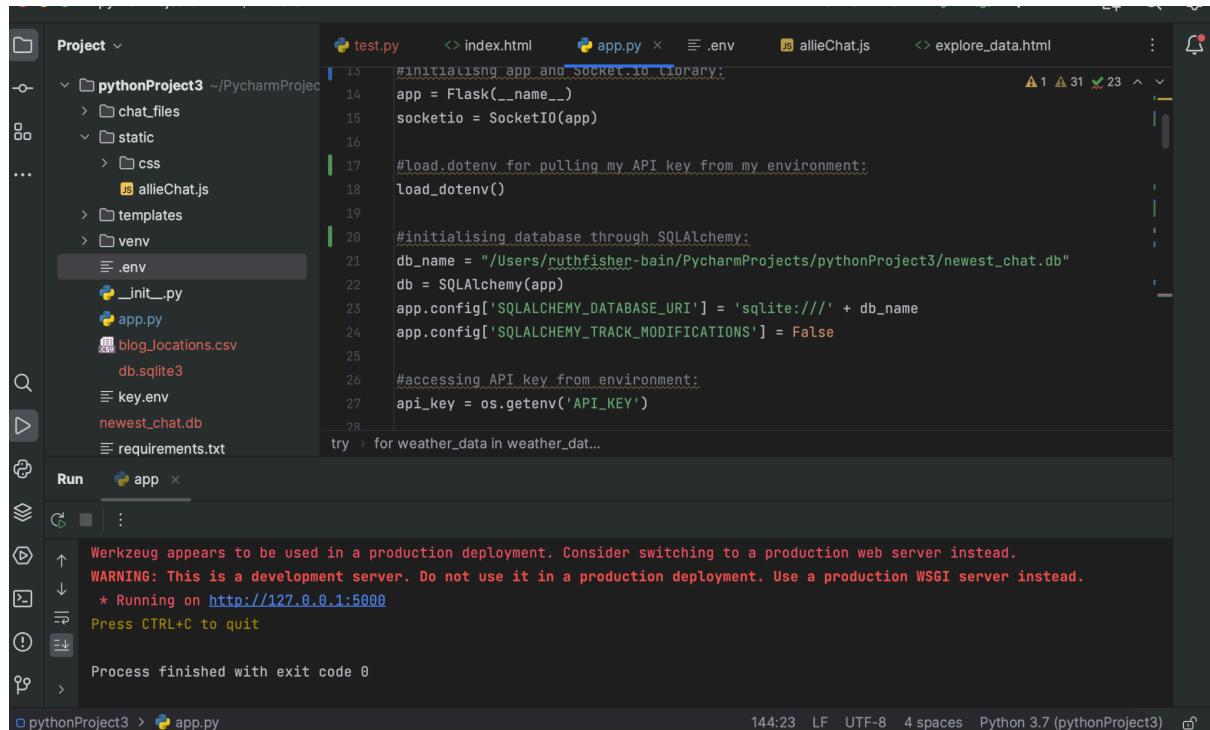
I added another two text files of example chatbot chat to chat\_files:



The screenshot shows the VS Code interface with the project 'pythonProject3' open. The 'app.py' file is the active tab, displaying Python code for training a chatbot. The code includes additional training data from 'locations.txt', 'locations2.txt', and 'weather.txt'. It also includes a section for training the bot using pandas and Open Weather Map data. The terminal below shows a log of requests from 127.0.0.1, indicating the bot is running and responding to socket.io connections.

```
"I'm AllieChat and I'm here for all your ExploreUK weather needs!"  
]  
  
list_trainer.train(list(intro for intro in introductions))  
  
list_trainer.train('chat_files/chats.txt')  
list_trainer.train('chat_files/weather.txt')  
list_trainer.train('chat_files/location.txt')  
list_trainer.train('chat_files/locations2.txt')  
  
corpus_trainer = ChatterBotCorpusTrainer(my_bot)  
corpus_trainer.train('chatterbot.corpus.english')  
corpus_trainer.train('chatterbot.corpus.english.conversations')  
  
# training the bot using pandas:  
logger_locations = pd.read_csv('weather_forecast.csv')  
try: # for loop to retrieve weather_forecast data:  
    for index, row in logger_locations.iterrows():  
        conditions = row['conditions']  
        list_trainer.train([conditions]) # error handling:  
except Exception as e:  
    print(f"Error training with data from weather_forecast.csv: {e}")  
  
# training bot using Open Weather Map data and my API key:
```

I configured ChatterBot with an SQLite storage adapter using the my\_bot instance, and after building a database (see “**Wrangling Data Using SQLite Database**”, page 18) I initialised the database through SQLAlchemy for Flask:



The screenshot shows the PyCharm IDE interface. On the left is the Project tool window displaying files like test.py, index.html, app.py, .env, and static/css/allieChat.js. The main editor window shows Python code for initializing a Flask app, setting up SocketIO, loading environment variables from .env, and initializing a SQLite database. The terminal window at the bottom shows the application running on port 5000, with a warning about Werkzeug being used in development mode.

```
#INITIALISING app and socket.io Library:
app = Flask(__name__)
socketio = SocketIO(app)

#load_dotenv for pulling my API key from my environment:
load_dotenv()

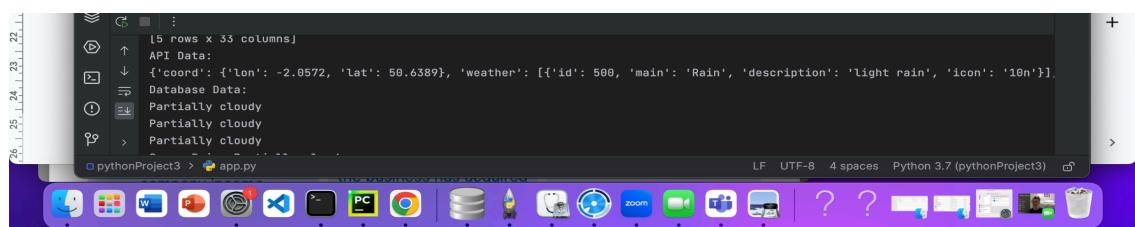
#initialising database through SQLAlchemy:
db_name = "/Users/ruthfisher-bain/PycharmProjects/pythonProject3/newest_chat.db"
db = SQLAlchemy(app)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///{}' + db_name
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

#accesing API key from environment:
api_key = os.getenv('API_KEY')

try > for weather_data in weather_dat...
```

```
Werkzeug appears to be used in a production deployment. Consider switching to a production web server instead.
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
Process finished with exit code 0
```

I confirmed the data was correctly being pulled from the SQL and other locations thanks to my js:



The screenshot shows a browser developer tools console window. It displays a JSON object representing weather data, with an expandable tree view showing nested properties like coordinates and weather descriptions.

```
15 rows x 33 columns
API Data:
{
  "coord": {"lon": -2.0572, "lat": 50.6389}, 
  "weather": [{"id": 500, "main": "Rain", "description": "light rain", "icon": "10n"}]
}
Database Data:
Partially cloudy
Partially cloudy
Partially cloudy
```

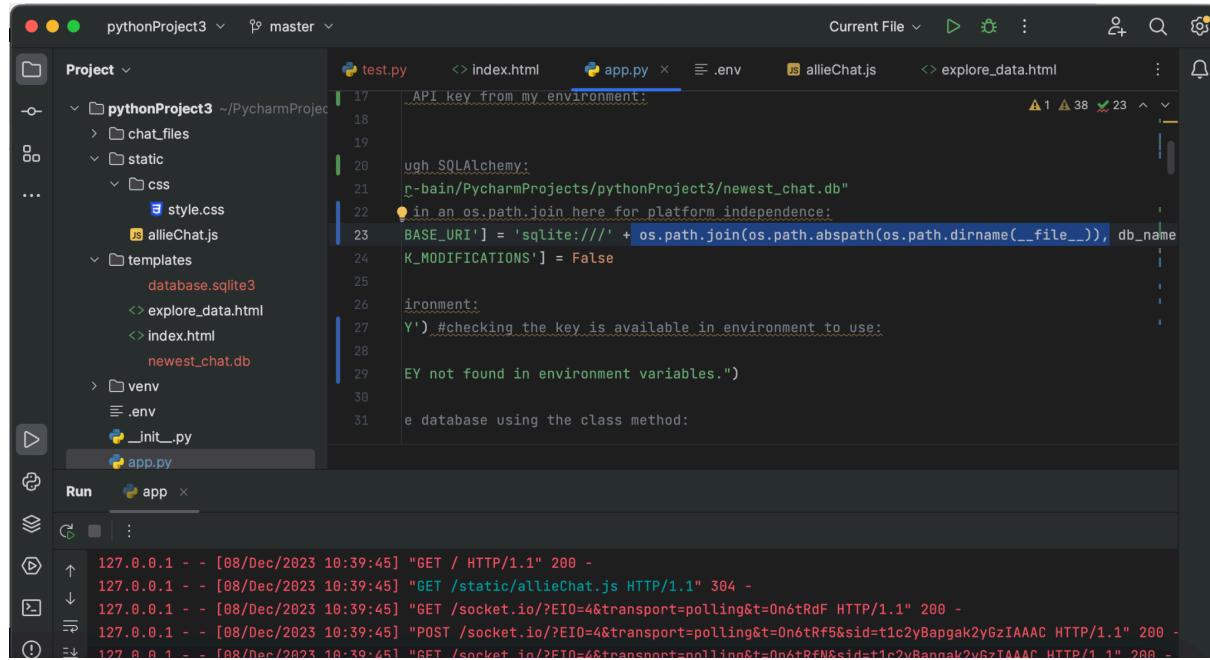
The screenshot shows a PyCharm interface with four main sections:

- Top Left:** A tree view showing the project structure with files like `Training computers.yml`, `Training conversations.yml`, `Training emotion.yml`, `Training food.yml`, `Training gossip.yml`, `Training greetings.yml`, and `Training health.yml`. Each file has a progress bar at 100% completion.
- Top Middle:** A code editor for `allieChat.js` containing CSV data for weather locations. The data includes columns: id, name, icon, and stations. Five entries for Birmingham are listed with icons like partly-cloudy-day, snow, and rain.
- Bottom Middle:** A code editor for `test.py` showing a function to fetch weather data from an API. It uses Axios to make a GET request to `https://api.openweathermap.org/data/2.5/weather?q=${location}&appid=${apiKey}`.
- Bottom Right:** A terminal window showing the output of a command. The output includes a warning about serving a Flask app in production mode and a note about using a WSGI server instead.

## 2. Python code to build the chatbot

## Building the Bot App

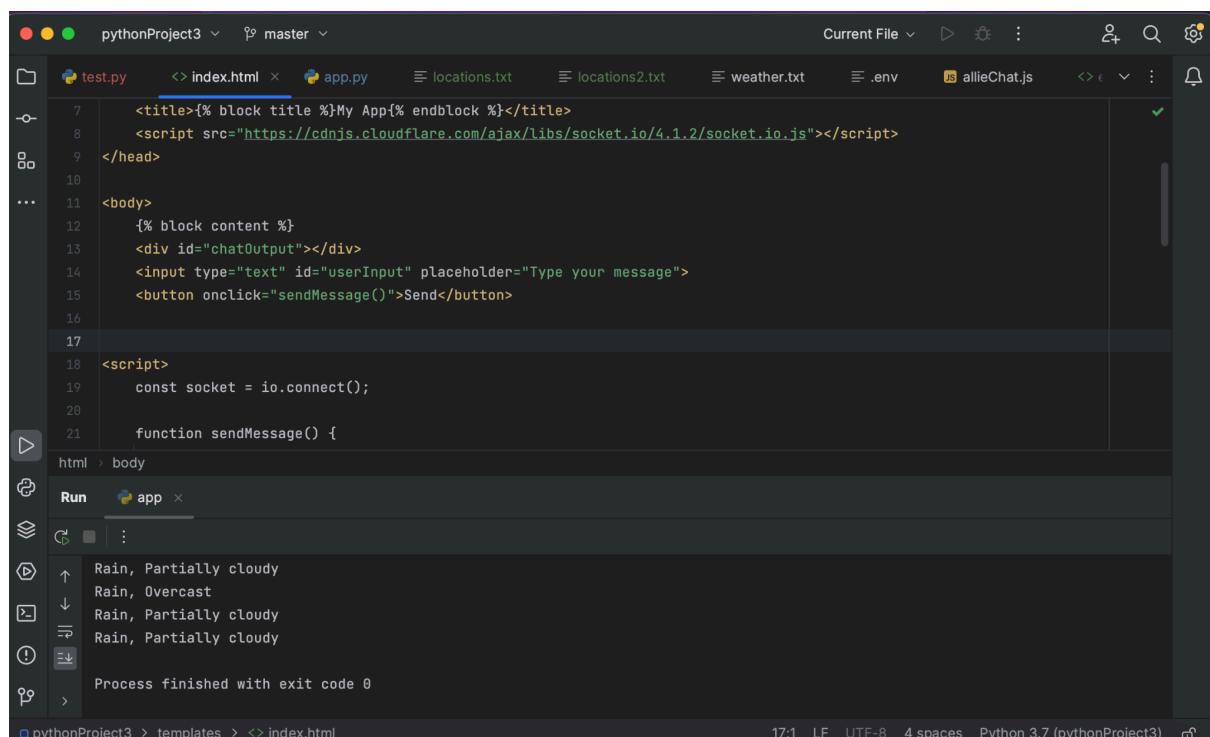
First, I added in an os.path.join function for platform independence:



The screenshot shows the PyCharm IDE interface. The project structure on the left includes a 'pythonProject3' folder containing 'chat\_files', 'static' (with 'css' and 'allieChat.js'), 'templates' (with 'database.sqlite3', 'explore\_data.html', 'index.html', and 'newest\_chat.db'), 'venv', '.env', '\_init\_.py', and 'app.py'. The 'app.py' file is open in the editor, showing Python code related to SQLAlchemy and environment variables. The terminal at the bottom shows a series of HTTP requests from '127.0.0.1' to the application, indicating it is running and responding to client requests.

```
API key from my environment:  
  17  
  18  
  19  
  20  ough SQLAlchemy:  
  21  p-bain/PycharmProjects/pythonProject3/newest_chat.db"  
  22  in an os.path.join here for platform independence:  
  23  BASE_URI] = 'sqlite:///` + os.path.abspath(os.path.dirname(__file__)), db_name  
  24  K_MODIFICATIONS] = False  
  25  
  26 ironment:  
  27 Y') #checking the key is available in environment to use:  
  28  
  29 EY not found in environment variables.")  
  30  
  31 e database using the class method:  
  
127.0.0.1 - - [08/Dec/2023 10:39:45] "GET / HTTP/1.1" 200 -  
127.0.0.1 - - [08/Dec/2023 10:39:45] "GET /static/allieChat.js HTTP/1.1" 304 -  
127.0.0.1 - - [08/Dec/2023 10:39:45] "GET /socket.io/?EIO=4&transport=polling&t=On6tRdF HTTP/1.1" 200 -  
127.0.0.1 - - [08/Dec/2023 10:39:45] "POST /socket.io/?EIO=4&transport=polling&t=On6tRf5&sid=t1c2yBapgak2yGzIAAAC HTTP/1.1" 200 -  
127.0.0.1 - - [08/Dec/2023 10:39:45] "GET /socket.io/?EIO=4&transport=polling&t=On6tRfNb&sid=t1c2yBapgak2yGzIAAAC HTTP/1.1" 200 -
```

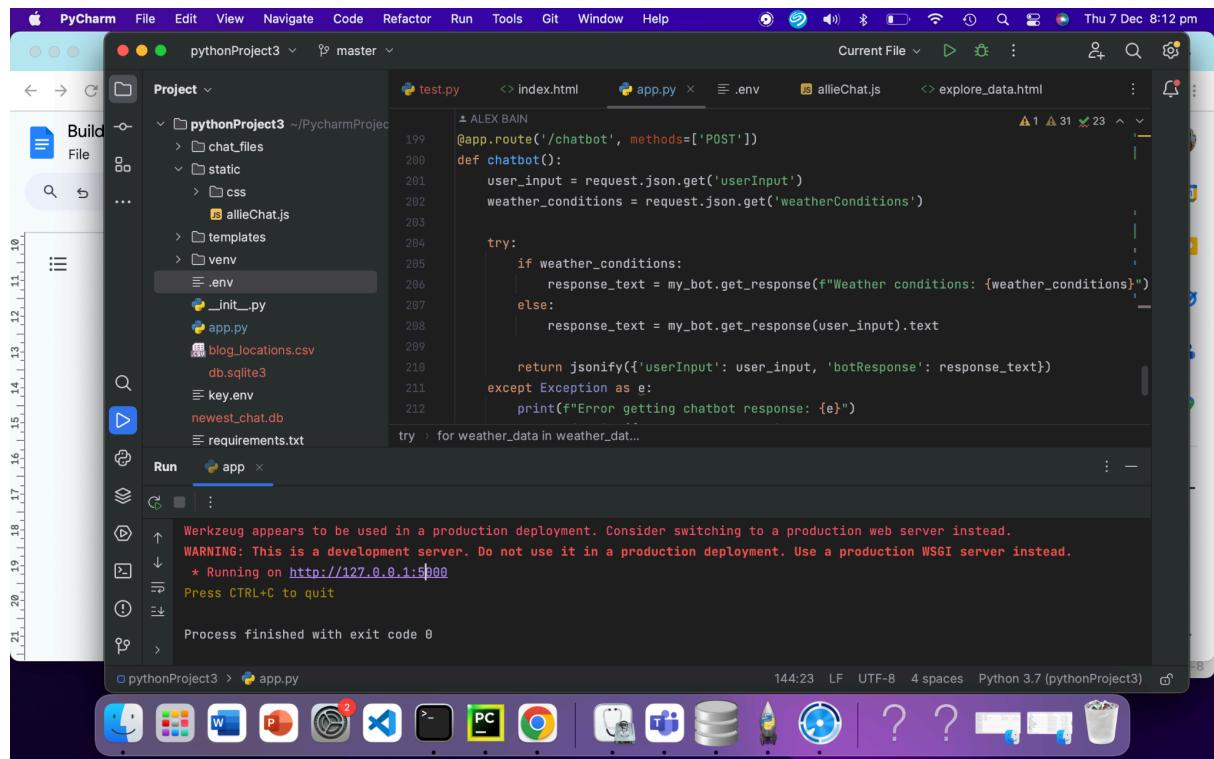
I wrote an index.html after doing some research on chatbot templates:



The screenshot shows the PyCharm IDE interface. The project structure on the left includes a 'pythonProject3' folder containing 'test.py', 'index.html', 'app.py', 'locations.txt', 'locations2.txt', 'weather.txt', '.env', 'allieChat.js', and 'templates' (with 'index.html'). The 'index.html' file is open in the editor, showing HTML and JavaScript code for a chatbot template. The terminal at the bottom shows the application's response to user input, displaying weather information for different locations.

```
<title>{% block title %}My App{% endblock %}</title>  
<script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.1.2/socket.io.js"></script>  
</head>  
...  
<body>  
  {% block content %}  
    <div id="chatOutput"></div>  
    <input type="text" id="userInput" placeholder="Type your message">  
    <button onclick="sendMessage()">Send</button>  
  </div>  
  <script>  
    const socket = io.connect();  
    ...  
    function sendMessage() {  
      const userMessage = document.getElementById('userInput').value;  
      const chatOutput = document.getElementById('chatOutput');  
      const messageElement = document.createElement('div');  
      messageElement.textContent = userMessage;  
      chatOutput.appendChild(messageElement);  
      socket.emit('user message', userMessage);  
    }  
    socket.on('server message', function(data) {  
      const chatOutput = document.getElementById('chatOutput');  
      const messageElement = document.createElement('div');  
      messageElement.textContent = data.message;  
      chatOutput.appendChild(messageElement);  
    });  
  </script>  
</body>  
  
Run app  
  
html > body  
Run app  
  
Rain, Partially cloudy  
Rain, Overcast  
Rain, Partially cloudy  
Rain, Partially cloudy  
Process finished with exit code 0
```

I wrote a basic chatbot route in app.py:



The screenshot shows the PyCharm IDE interface. The project navigation bar at the top indicates the current file is app.py. The code editor displays Python code for a chatbot route. The terminal below shows the application running on a local server.

```
ALEX BAIN
@app.route('/chatbot', methods=['POST'])
def chatbot():
    user_input = request.json.get('userInput')
    weather_conditions = request.json.get('weatherConditions')

    try:
        if weather_conditions:
            response_text = my_bot.get_response(f"Weather conditions: {weather_conditions}")
        else:
            response_text = my_bot.get_response(user_input).text

        return jsonify({'userInput': user_input, 'botResponse': response_text})
    except Exception as e:
        print(f"Error getting chatbot response: {e}")

try:
    for weather_data in weather_dat...
```

Terminal output:

```
Werkzeug appears to be used in a production deployment. Consider switching to a production web server instead.
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
Process finished with exit code 0
```

I implemented a retrain-chatbot route to train the bot as it collected user input and output responses:

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top indicates the current file is `app.py`. The code editor displays Python code for a chatbot, specifically a function to retrain it with weather data. A terminal window below the editor shows the output of running the application on port 5000, including a warning about Werkzeug being used in a development server. The status bar at the bottom right shows the Python version is 3.7.

```
def retrain_chatbot():
    try:
        all_weather_data = Data.query.all()
        for item in all_weather_data:
            list_trainer.train([item.conditions])

        print("Chatbot retrained successfully with weather data.")
    except Exception as e:
        print(f"Error retraining with data from weather_table: {e}")

#defining_chatbot_response:
def get_chatbot_response(user_input, weather_conditions=None):
    try:
        for weather_data in weather_dat...
```

However, as the chatbot was slow, and training can still work without explicitly retraining it every run, I deleted this function, and instead focused on sufficiently training the bot without it.

I defined the `get_chatbot_response`:

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top indicates the current file is `app.py`. The code editor displays Python code for defining the `get_chatbot_response` function. A tooltip is visible over the `Exception` class definition, providing documentation from the Python documentation website. The status bar at the bottom right shows the Python version is 3.7.

```
#defining_chatbot_response:
def get_chatbot_response(user_input, weather_conditions=None):
    try:
        if weather_conditions:
            response = my_bot.get_response(f"Weather conditions: {weather_conditions}")
        else:
            response = my_bot.get_response(user_input)
        return response.text
    except Exception as e:
        print(f"Er...builtins")
        return "I...class Exception(BaseException)
Common base class for all non-exit exceptions.
'Exception(BaseException)' on docs.python.org..."
```

This helps to modularize my code and makes it easier to reuse the chatbot logic in different parts of the app.

Along with this and the app.route for chatbot, I created a Socket.io library route for the handling of the chatbot message, before running the app:

The screenshot shows the PyCharm IDE interface. The top bar displays the project name 'pythonProject3' and branch 'master'. The left sidebar shows the project structure with files like 'app.py', 'test.py', 'index.html', 'corpus.py', 'chatbot.html', 'allieChat.js', 'base.html', 'css', 'static', 'templates', 'venv', and 'db.sqlite3'. The main code editor window is open on 'app.py' and contains the following code:

```
187     @socketio.on('message')
188     def handle_message(msg):
189         response = get_chatbot_response(msg)
190         socketio.emit('message', {'user': msg, 'bot': response})
191
192     # Running my app through SocketIO:
193     if __name__ == '__main__':
194         try:
195             print(f"Database file path: {app.config['SQLALCHEMY_DATABASE_URI']}")
196             db.create_all()
197             socketio.run(app, debug=True, allow_unsafe_werkzeug=True, use_reloader=False)
198         except Exception as e:
199             print(f"Error creating database or running app: {e}")
200
201     chatbot() > if request.method == 'POST'
```

The Python Console window below shows an error message:

```
File "/Applications/PyCharm CE.app/Contents/plugins/python-ce/helpers/pydev/pydevconsole.py", line 25, in <module>
    from _pydev_bundle.pydev_imports import _queue
File "/Applications/PyCharm CE.app/Contents/plugins/python-ce/helpers/pydev/_pydev_bundle/pydev_imports.py", line 39, in <module>
    from _pydev_imps._pydev_execfile import execfile
ImportError: cannot import name 'execfile' from '_pydev_imps._pydev_execfile' (/Applications/PyCharm CE.app/Contents/plugins/python-ce/help
... Couldn't connect to console process.
```

The bottom status bar indicates the file is 'app.py', encoding is 'UTF-8', and the Python version is '3.7 (py)'.

## Pulling Data from OpenWeatherMaps API

As the current prototype can only return one location at a time, I have implemented an API through a function that calls to several locations at once and can collect information from multiple locations at the same time. I implemented a list for the function to loop over, along with my new API key:

```

113 def fetch_openweathermap_data_for_multiple_cities(cities):
114     weather_data_list = []
115
116     for city in cities:
117         base_url = 'http://api.openweathermap.org/data/2.5/weather'
118         params = {
119             'q': city,
120             'appid': api_key,
121         }
122
123         try:
124             response = requests.get(base_url, params=params)
125             response.raise_for_status()
126             weather_data = response.json()
127             description = weather_data.get('weather')[0].get('description')
128
129             weather_data_list.append({'city': city, 'description': description})
130
131         except requests.exceptions.RequestException as e:
132             print(f"Error fetching OpenWeather data for {city}: {e}")
133
134     # List of cities to fulfill customer needs of returning more than one location at a time:
135     city_names = ['CorfeCastle', 'TheCotswolds', 'Bristol', 'Oxford', 'Norwich', 'Stonehenge',
136                  'WatergateBay', 'Birmingham']
137
138     return weather_data_list

```

Werkzeug appears to be used in a production deployment. Consider switching to a production web server instead.  
 WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
 \* Running on <http://127.0.0.1:5000>  
 Press CTRL+C to quit

Despite deleting the `retrain_bot` function, AllieChat was still slow. I researched and imported `ThreadPoolExecutor` with `parallel` - and therefore more efficient- processing:

The screenshot shows the PyCharm IDE interface. The project structure on the left includes files like `chat_files`, `static`, `templates`, `venv`, `.env`, `__init__.py`, `app.py`, `blog_locations.csv`, `db.sqlite3`, `key.env`, `newest_chat.db`, `newest_chat.db-shm`, `newest_chat.db-wal`, and `requirements.txt`. The code editor window displays `app.py` with code for fetching OpenWeather data in parallel using `ThreadPoolExecutor`. The run output window shows a warning about Werkzeug being used in production mode and a log entry indicating the application is running on port 127.0.0.1:5000.

```
# Function to fetch OpenWeather data for a single city
base_url = 'http://api.openweathermap.org/data/2.5/weather'
params = {'q': city, 'appid': api_key}
response = requests.get(base_url, params=params)
response.raise_for_status()
return response.json()

# List of cities to fetch OpenWeather data for
city_names = ['CorfeCastle', 'TheCotswolds', 'Bristol', 'Oxford', 'Norwich', 'Stonehenge', 'WatergateBay']

# Fetch OpenWeather data in parallel
with ThreadPoolExecutor() as executor:
    weather_data_list = list(executor.map(fetch_openweather_data, city_names))
```

Thanks to the `ThreadPoolExecutor()` as executor the function now fetched OpenWeather data in parallel with the `weather_data_list`.

The Cotswolds and Watergate Bay returned errors, so instead I called from Gloucestershire and Newquay:

The screenshot shows the PyCharm IDE interface. The project structure on the left includes files like `explore_data.html`, `index.html`, `newest_chat.db`, `venv`, `.env`, `__init__.py`, `app.py`, `blog_locations.csv`, `db.sqlite3`, `newest_chat.db`, `requirements.txt`, `test.py`, and `weather_forecast.csv`. The code editor window displays `test.py` with a modified list of locations. The run output window shows an error message indicating a 404 Client Error for the location 'WatergateBay'.

```
def fetch_openweather_data(location):
    # Function to fetch OpenWeather data for a single location
    base_url = 'http://api.openweathermap.org/data/2.5/weather'
    params = {'q': location, 'appid': api_key}
    response = requests.get(base_url, params=params)
    response.raise_for_status()
    return response.json()

# List of locations to fetch OpenWeather data for:
location_name = ['Corfe Castle', 'Gloucestershire', 'Cambridge',
                  'Stonehenge', 'Newquay', 'Corfe Castle',
                  'Bristol', 'Oxford', 'Norwich']
```

## Pulling Data through CSV and Pandas

To pull data from the csv file, “weather\_data”, using Pandas, I first worked on data retrieval and reading using Pandas to test my data.

First, I created a csv file (see **Wrangling Data Using DB Browser and SQLite**, page 18). Using pd.read\_csv to import itinerary.csv, and head() and columns() to display, I retrieved data using python Object-Oriented Programming methods in a separate py file for testing:

```
#Importing the data sets:  
blogger_locations = pd.read_csv('blog_locations.csv')  
print(blogger_locations.head())  
print(blogger_locations.columns)  
  
birmingham_weather = blogger_locations[(blogger_locations['name'] == 'Birmingham') &  
(blogger_locations['datetime'] == '2023-11-29')]  
  
# Viewing the output:  
print(birmingham_weather)  
  
# Exploring the data.  
# Any missing values?  
print(blogger_locations.isna().sum())  
  
# Viewing the metadata:  
print(blogger_locations.info())  
  
app = Flask(__name__)  
  
@app.route('/weather/<city>/<date>')  
def display_weather(city, date):  
    # Filter weather data based on the requested city and date  
    city_weather = blogger_locations[(blogger_locations['name'] == city) &  
(blogger_locations['datetime'] == date)]  
  
    if not city_weather.empty:  
        # Pass relevant information to the template  
        return render_template('weather_template.html', city=city, date=date,  
temperature=city_weather.iloc[0]['temp'], conditions=city_weather.iloc[0]['conditions'])  
    else:  
        return render_template('weather_not_found.html', city=city, date=date)  
  
if __name__ == '__main__':
```

```
app.run(debug=True)
```

The screenshot shows the PyCharm interface with the code editor open to `app.py`. The terminal window at the bottom displays the output of the application. The application is running in debug mode and is printing weather forecast data for Birmingham. The data is shown in two parts: a full dataset and a single row summary.

```
37     app.run(debug=True)
if __name__ == '__main__':
    Press CTRL+C to quit
    * Restarting with stat
    name      datetime ...      icon      stations
0  Birmingham  2023-11-29 ...  partly-cloudy-day  EGBB,EGOS,EGNX,AU757
1  Birmingham  2023-11-30 ...  partly-cloudy-day      NaN
2  Birmingham  2023-12-01 ...  partly-cloudy-day      NaN
3  Birmingham  2023-12-02 ...  partly-cloudy-day      NaN
4  Birmingham  2023-12-03 ...        snow      NaN
[5 rows x 33 columns]
Index(['name', 'datetime', 'tempmax', 'tempmin', 'temp', 'feelslikemax',
       'feelslikemin', 'feelslike', 'dew', 'humidity', 'precip', 'precipprob',
       'precipcover', 'preciptype', 'snow', 'snowdepth', 'windgust',
       'windspeed', 'winddir', 'sealevelpressure', 'cloudcover', 'visibility',
       'solarradiation', 'solarenergy', 'uvindex', 'severerisk', 'sunrise',
       'sunset', 'moonphase', 'conditions', 'description', 'icon', 'stations'],
      dtype='object')
      name      datetime ...      icon      stations
0  Birmingham  2023-11-29 ...  partly-cloudy-day  EGBB,EGOS,EGNX,AU757
[1 rows x 33 columns]
```

This screenshot shows another session of PyCharm with the same `app.py` file open. The terminal window displays the same weather forecast data for Birmingham, indicating that the application is running again due to the debug mode.

```
18
19     # Viewing the metadata:
20     print(blogger_locations.info())
21
22     app = Flask(__name__)
23
24
25
26
27     @app.route('/weather/<city>/<date>')
```

I implemented the dataframes into the app by calling from `blogger_locations = pd.read_csv('weather_forecast.csv')` at the top of the file, and a try-except error block:

```

corpus_trainer = ChatterBotCorpusTrainer(my_bot)
corpus_trainer.train('chatterbot.corpus.english')
corpus_trainer.train('chatterbot.corpus.english.conversations')

#training the bot using pandas:
logger_locations = pd.read_csv('weather_forecast.csv')
try: #for loop to retrieve weather_forecast data:
    for index, row in logger_locations.iterrows():
        conditions = row['conditions']
        list_trainer.train([conditions]) #error handling:
except Exception as e:
    print(f"Error training with data from weather_forecast.csv: {e}")

```

## Wrangling Data Using DB Browser and SQLite

To solve the company's issue of only returning daily weather data, I used Visual Crossing global weather database's Weather Query Builder. I created a csv file by inputting the locations on the blogger's itinerary over a two-week period from a future date (note: the location Lake District here, but this wasn't specifically called in the app):

The screenshot shows the Visual Crossing Weather Query Builder interface. At the top, there are navigation links for Weather Data, Weather API, Query Builder, Pricing, API Docs, and More, along with a search bar and user account information. Below the header, the title "Weather Query Builder" is displayed, along with a "Legacy version" link. The main query area includes fields for location ("Lakes District, England, United Kingdom"), time range ("Next 15 days", "29/11/2023", "13/12/2023"), metric ("Metric (°C, km)"), and data units ("Data units"). There are also sections for "Data sections" (Weather elements, Degree days, Wind & solar, Agriculture, Weather stations), "Additional Data" (Daily, Hourly, Current, Events, Alerts), "Data Details" (Info, Stations), and download options (API, Grid, Chart, JSON, CSV). A "Download" button is located at the bottom right.

**Processing**

Content type: csv, Output sections: days  
 Location count: 10, Estimated record count: 60  
 Lake District England 2023-12-12 to 2023-12-19

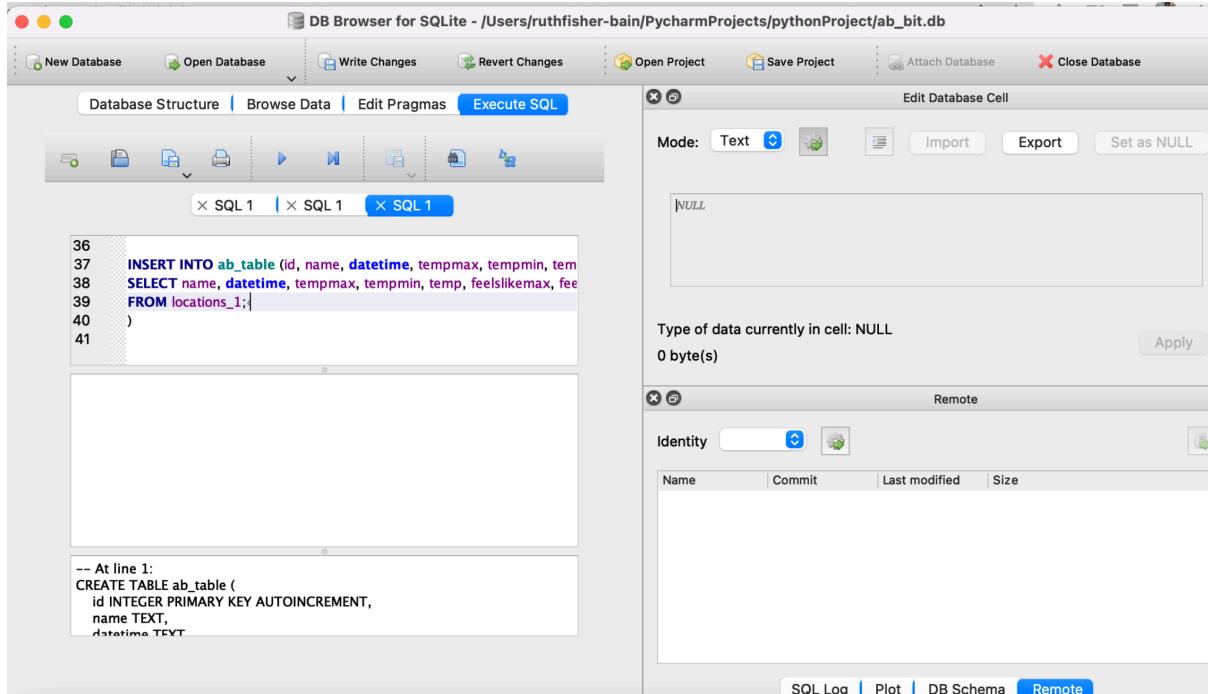
Submitting...

60% completed

- ✓ Data format and section
- ✓ Dataset name
- ✓ Submit or schedule
- ✗ Processing
- ✗ Download

DB Browser for SQLite helped me identify how the different components of the app and database worked together, and to more clearly decipher any issues I had or errors I received.

Once I downloaded the csv, a relational database was easy to logically design from my initial concept through DBMS SQLite, ensuring accuracy, consistency and durability. Physically, this was more challenging. I added the id heading into the table, as this wasn't in the original csv code, via commands, incorporated this into the app, and investigated errors carefully. Data was displayed from both weather\_table and locations\_data:



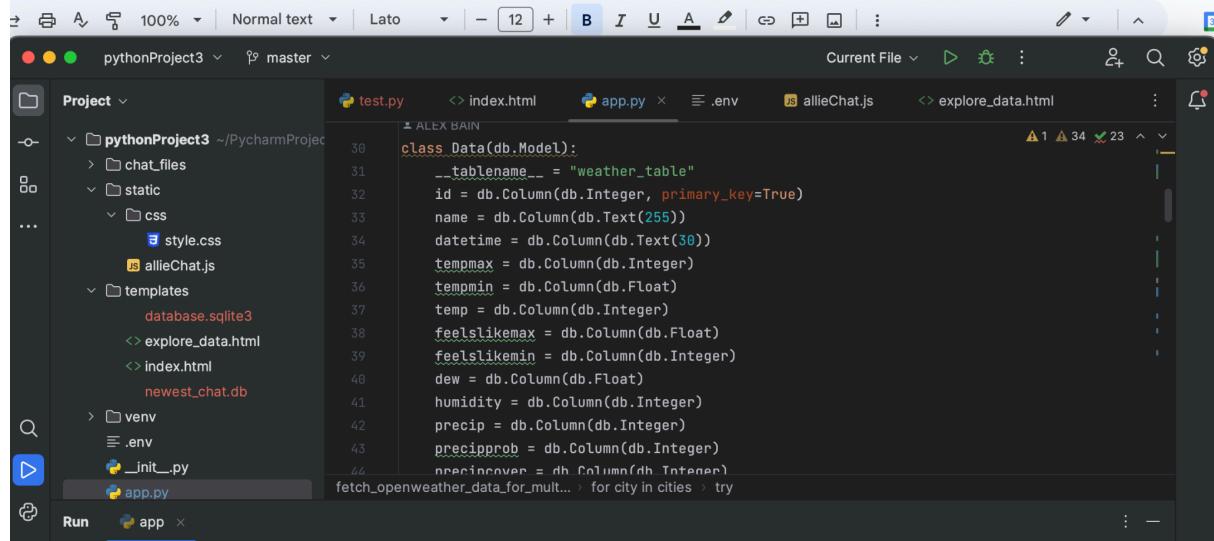
I added in an exception to ensure that I was retrieving data correctly, which helped heaps when I realised I had not completed the Write of the newest\_chat.db database:

```
Exception as e:  
    print(f"Error creating database: {e}")  
    # Add the following line to print the stack trace  
    import traceback  
    traceback.print_exc()
```

Eventually, I was able to replicate the database in my app via an “index.html” template, and this told me that the app and browser were reading the data correctly.

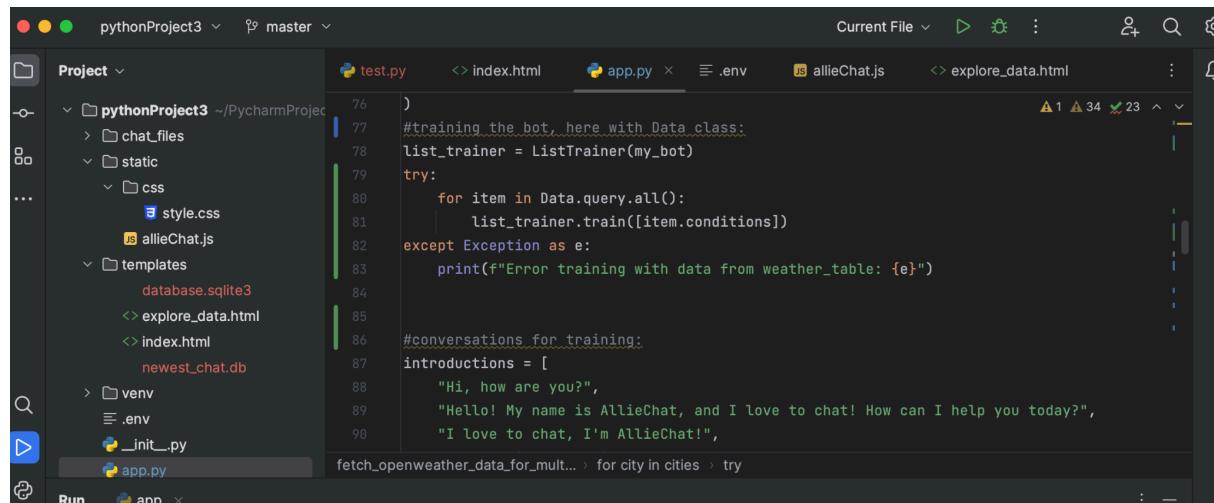
## Calling Data from Data Class:

Now that my database had been properly created, I created a Data class in my app.py file:



```
ALEX BAIN
class Data(db.Model):
    __tablename__ = "weather_table"
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.Text(255))
    datetime = db.Column(db.Text(30))
    tempmax = db.Column(db.Integer)
    tempmin = db.Column(db.Float)
    temp = db.Column(db.Integer)
    feelslikemax = db.Column(db.Float)
    feelslikemin = db.Column(db.Integer)
    dew = db.Column(db.Float)
    humidity = db.Column(db.Integer)
    precip = db.Column(db.Integer)
    precipprob = db.Column(db.Integer)
    nprecincover = db.Column(db.Integer)
    fetch_openweathermap_data_for_mult... > for city in cities > try
```

I called trained the bot using ListTrainer and data from Data class:



```
76 )
77 #training the bot, here with Data class:
78 list_trainer = ListTrainer(my_bot)
79 try:
80     for item in Data.query.all():
81         list_trainer.train([item.conditions])
82 except Exception as e:
83     print(f"Error training with data from weather_table: {e}")
84
85 #conversations for training:
86 introductions = [
87     "Hi, how are you?",
88     "Hello! My name is AllieChat, and I love to chat! How can I help you today?",
89     "I love to chat, I'm AllieChat!",
90
91     fetch_openweathermap_data_for_mult... > for city in cities > try
```

I added in db.create\_all() to create database tables as good code practice is placed above the app's run statement, which includes debugging and calls from Socket.IO:

The screenshot shows the PyCharm IDE interface. The project structure on the left includes a 'pythonProject3' folder containing 'chat\_files', 'static', 'templates', 'venv', and several Python files like '\_init\_.py', 'app.py', and 'test.py'. The 'app.py' tab is active, displaying code related to a Flask application and SocketIO. The terminal window at the bottom shows a series of HTTP requests from '127.0.0.1' to the application, indicating it is running.

```

try:
    # Print the database file path for debugging purposes
    print(f"Database file path: {app.config['SQLALCHEMY_DATABASE_URI']}")

    # Creating the database tables- this was a huge step I had initially missed!
    db.create_all()

    # Run the Flask app with SocketIO
    socketio.run(app, debug=True, allow_unsafe_werkzeug=True, use_reloader=False)

except Exception as e:
    print(f"Error creating database or running app: {e}")
    import traceback
    traceback.print_exc()

```

## Chatbot Javascript

My initial simple JS code was slow or unresponsive. After a fair bit of testing in a test.py file, I sought some advice from Gita, and realised the event listener I'd called from js wasn't receiving the request. My script wasn't sending the user input to the Flask backend. I had forgotten to replace the placeholder string with a HTTP POST request to a server endpoint, which I amended using the Axios library:

The screenshot shows the PyCharm IDE interface. The project structure on the left includes a 'pythonProject3' folder containing 'chat\_files', 'static', 'templates', 'venv', and several Python files like '\_init\_.py', 'app.py', and 'test.py'. The 'allieChat.js' tab is active, displaying JavaScript code that uses the Axios library to send POST requests to a '/chatbot' endpoint. A yellow lightbulb icon is visible on line 56, likely indicating a warning or suggestion.

```

server.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});

// Function to get a response from my chatterbot:
function getChatbotResponse(userInput) {
    // Sending the user input to the Flask backend, another important step I missed!
    axios.post('/chatbot', { userInput: userInput })
        .then(response => {
            const botResponse = response.data.botResponse;
            // Processing the bot response:
            console.log('Bot Response:', botResponse);
        })
        .catch(error => {
            console.error('Error getting bot response:', error);
        });
}

```

I tested chatbot output in a separate project, and uninstalled then reinstalled Socket.IO . I added in the static path and socket.io requirement, which were initially missing in my js:

```

13 // using static files from the "static" directory
14 app.use(express.static('static'));
15
16 // Define a route to serve your HTML file
17 app.get('/', (req, res) => {
18   res.sendFile(__dirname + '/templates/index.html');
19 });
20
21 // Event listener for incoming connections
22 io.on('connection', (socket) => {
23   console.log('A user connected');
24
25   // Event listener for incoming messages:
26   socket.on('message', (data) => {
27     // Your existing message handling logic
28     console.log(`User ${socket.id} - ${data.message}`);
29   });
30 });
31
32 // Start the server
33 const PORT = process.env.PORT || 3001;
34 app.listen(PORT, () => {
35   console.log(`Server is running on port ${PORT}`);
36 });

```

My output showed I had successfully established WebSocket connections via Socket.IO. I ensured my index.html called the chatbot properly, including JS and Socket.IO links:

```

36
37   document.getElementById("chatOutput").innerHTML +=
38     '<div><strong>User:</strong> ' + data.user_input + '</div>' +
39     '<div><strong>Bot:</strong> ' + data.bot_response + '</div>';
40   });
41 </script>
42 {%- endblock %}
43
44 <script src="{{ url_for('static', filename='allieChat.js') }}"></script>
45 <script>
46   var socket = io.connect('http://' + document.domain + ':' + location.port);
47 </script>
48 </body>
49 </html>

```

```

3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>{% block title %}My App{% endblock %}</title>
7     <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.1.2/socket.io.js"></script>
8 </head>
9
10 <body>
11     {% block content %}
12         <div id="chatOutput"></div>
13         <input type="text" id="userInput" placeholder="Type your message">
14         <button onclick="sendMessage()">Send</button>
15
16     <script>
17         html > body

```

I added some more error handling to my functions including the Exception method to check I was pulling the correct data:

```

140 # Training the chatbot with the Open Weather data:
141 try:
142     for weather_data in weather_data_list:
143         description = weather_data.get('description')
144         if description:
145             list_trainer.train([description])
146     except Exception as e:
147         print(f"Error training with OpenWeather data: {e}")
148
149 #retrain_chatbot function to consistently train AllieChat:
150
151
152 def retrain_chatbot():
153     try:
154         ...
155     finally:
156         ...
157
158     ...
159
160     ...
161
162     ...
163
164     ...
165
166     ...
167
168     ...
169
170     ...
171
172     ...
173
174     ...
175
176     ...
177
178     ...
179
180     ...
181
182     ...
183
184     ...
185
186     ...
187
188     ...
189
190     ...
191
192     ...
193
194     ...
195
196     ...
197
198     ...
199
200     ...
201
202     ...
203
204     ...
205
206     ...
207
208     ...
209
210     ...
211
212     ...
213
214     ...
215
216     ...
217
218     ...
219
220     ...
221
222     ...
223
224     ...
225
226     ...
227
228     ...
229
230     ...
231
232     ...
233
234     ...
235
236     ...
237
238     ...
239
240     ...
241
242     ...
243
244     ...
245
246     ...
247
248     ...
249
250     ...
251
252     ...
253
254     ...
255
256     ...
257
258     ...
259
260     ...
261
262     ...
263
264     ...
265
266     ...
267
268     ...
269
270     ...
271
272     ...
273
274     ...
275
276     ...
277
278     ...
279
280     ...
281
282     ...
283
284     ...
285
286     ...
287
288     ...
289
290     ...
291
292     ...
293
294     ...
295
296     ...
297
298     ...
299
299
300     ...
301
302     ...
303
304     ...
305
306     ...
307
308     ...
309
309
310     ...
311
312     ...
313
314     ...
315
316     ...
317
317
318     ...
319
320     ...
321
322     ...
323
323
324     ...
325
326     ...
327
328     ...
329
329
330     ...
331
332     ...
333
334     ...
335
335
336     ...
337
338     ...
339
339
340     ...
341
342     ...
343
344     ...
345
345
346     ...
347
348     ...
349
349
350     ...
351
352     ...
353
354     ...
355
355
356     ...
357
358     ...
359
359
360     ...
361
362     ...
363
364     ...
365
365
366     ...
367
368     ...
369
369
370     ...
371
372     ...
373
374     ...
375
375
376     ...
377
378     ...
379
379
380     ...
381
382     ...
383
384     ...
385
385
386     ...
387
388     ...
389
389
390     ...
391
392     ...
393
394     ...
395
395
396     ...
397
398     ...
399
399
400     ...
401
402     ...
403
404     ...
405
405
406     ...
407
408     ...
409
409
410     ...
411
412     ...
413
414     ...
415
415
416     ...
417
418     ...
419
419
420     ...
421
422     ...
423
424     ...
425
425
426     ...
427
428     ...
429
429
430     ...
431
432     ...
433
434     ...
435
435
436     ...
437
438     ...
439
439
440     ...
441
442     ...
443
444     ...
445
445
446     ...
447
448     ...
449
449
450     ...
451
452     ...
453
454     ...
455
455
456     ...
457
458     ...
459
459
460     ...
461
462     ...
463
464     ...
465
465
466     ...
467
468     ...
469
469
470     ...
471
472     ...
473
474     ...
475
475
476     ...
477
478     ...
479
479
480     ...
481
482     ...
483
484     ...
485
485
486     ...
487
488     ...
489
489
490     ...
491
492     ...
493
494     ...
495
495
496     ...
497
498     ...
499
499
500     ...
501
502     ...
503
504     ...
505
505
506     ...
507
508     ...
509
509
510     ...
511
512     ...
513
514     ...
515
515
516     ...
517
518     ...
519
519
520     ...
521
522     ...
523
524     ...
525
525
526     ...
527
528     ...
529
529
530     ...
531
532     ...
533
534     ...
535
535
536     ...
537
538     ...
539
539
540     ...
541
542     ...
543
544     ...
545
545
546     ...
547
548     ...
549
549
550     ...
551
552     ...
553
554     ...
555
555
556     ...
557
558     ...
559
559
560     ...
561
562     ...
563
564     ...
565
565
566     ...
567
568     ...
569
569
570     ...
571
572     ...
573
574     ...
575
575
576     ...
577
578     ...
579
579
580     ...
581
582     ...
583
584     ...
585
585
586     ...
587
588     ...
589
589
590     ...
591
592     ...
593
594     ...
595
595
596     ...
597
598     ...
599
599
600     ...
601
602     ...
603
604     ...
605
605
606     ...
607
608     ...
609
609
610     ...
611
612     ...
613
614     ...
615
615
616     ...
617
618     ...
619
619
620     ...
621
622     ...
623
624     ...
625
625
626     ...
627
628     ...
629
629
630     ...
631
632     ...
633
634     ...
635
635
636     ...
637
638     ...
639
639
640     ...
641
642     ...
643
644     ...
645
645
646     ...
647
648     ...
649
649
650     ...
651
652     ...
653
654     ...
655
655
656     ...
657
658     ...
659
659
660     ...
661
662     ...
663
664     ...
665
665
666     ...
667
668     ...
669
669
670     ...
671
672     ...
673
674     ...
675
675
676     ...
677
678     ...
679
679
680     ...
681
682     ...
683
684     ...
685
685
686     ...
687
688     ...
689
689
690     ...
691
692     ...
693
694     ...
695
695
696     ...
697
698     ...
699
699
700     ...
701
702     ...
703
704     ...
705
705
706     ...
707
708     ...
709
709
710     ...
711
712     ...
713
714     ...
715
715
716     ...
717
718     ...
719
719
720     ...
721
722     ...
723
724     ...
725
725
726     ...
727
728     ...
729
729
730     ...
731
732     ...
733
734     ...
735
735
736     ...
737
738     ...
739
739
740     ...
741
742     ...
743
744     ...
745
745
746     ...
747
748     ...
749
749
750     ...
751
752     ...
753
754     ...
755
755
756     ...
757
758     ...
759
759
760     ...
761
762     ...
763
764     ...
765
765
766     ...
767
768     ...
769
769
770     ...
771
772     ...
773
774     ...
775
775
776     ...
777
778     ...
779
779
780     ...
781
782     ...
783
784     ...
785
785
786     ...
787
788     ...
789
789
790     ...
791
792     ...
793
794     ...
795
795
796     ...
797
798     ...
799
799
800     ...
801
802     ...
803
804     ...
805
805
806     ...
807
808     ...
809
809
810     ...
811
812     ...
813
814     ...
815
815
816     ...
817
818     ...
819
819
820     ...
821
822     ...
823
824     ...
825
825
826     ...
827
828     ...
829
829
830     ...
831
832     ...
833
834     ...
835
835
836     ...
837
838     ...
839
839
840     ...
841
842     ...
843
844     ...
845
845
846     ...
847
848     ...
849
849
850     ...
851
852     ...
853
854     ...
855
855
856     ...
857
858     ...
859
859
860     ...
861
862     ...
863
864     ...
865
865
866     ...
867
868     ...
869
869
870     ...
871
872     ...
873
874     ...
875
875
876     ...
877
878     ...
879
879
880     ...
881
882     ...
883
884     ...
885
885
886     ...
887
888     ...
889
889
890     ...
891
892     ...
893
894     ...
895
895
896     ...
897
898     ...
899
899
900     ...
901
902     ...
903
904     ...
905
905
906     ...
907
908     ...
909
909
910     ...
911
912     ...
913
914     ...
915
915
916     ...
917
918     ...
919
919
920     ...
921
922     ...
923
924     ...
925
925
926     ...
927
928     ...
929
929
930     ...
931
932     ...
933
934     ...
935
935
936     ...
937
938     ...
939
939
940     ...
941
942     ...
943
944     ...
945
945
946     ...
947
948     ...
949
949
950     ...
951
952     ...
953
954     ...
955
955
956     ...
957
958     ...
959
959
960     ...
961
962     ...
963
964     ...
965
965
966     ...
967
968     ...
969
969
970     ...
971
972     ...
973
974     ...
975
975
976     ...
977
978     ...
979
979
980     ...
981
982     ...
983
984     ...
985
985
986     ...
987
988     ...
989
989
990     ...
991
992     ...
993
994     ...
995
995
996     ...
997
998     ...
999
999
1000     ...
1001
1002     ...
1003
1004     ...
1005
1005
1006     ...
1007
1008     ...
1009
1009
1010     ...
1011
1012     ...
1013
1014     ...
1015
1015
1016     ...
1017
1018     ...
1019
1019
1020     ...
1021
1022     ...
1023
1024     ...
1025
1025
1026     ...
1027
1028     ...
1029
1029
1030     ...
1031
1032     ...
1033
1034     ...
1035
1035
1036     ...
1037
1038     ...
1039
1039
1040     ...
1041
1042     ...
1043
1044     ...
1045
1045
1046     ...
1047
1048     ...
1049
1049
1050     ...
1051
1052     ...
1053
1054     ...
1055
1055
1056     ...
1057
1058     ...
1059
1059
1060     ...
1061
1062     ...
1063
1064     ...
1065
1065
1066     ...
1067
1068     ...
1069
1069
1070     ...
1071
1072     ...
1073
1074     ...
1075
1075
1076     ...
1077
1078     ...
1079
1079
1080     ...
1081
1082     ...
1083
1084     ...
1085
1085
1086     ...
1087
1088     ...
1089
1089
1090     ...
1091
1092     ...
1093
1094     ...
1095
1095
1096     ...
1097
1098     ...
1099
1099
1100     ...
1101
1102     ...
1103
1104     ...
1105
1105
1106     ...
1107
1108     ...
1109
1109
1110     ...
1111
1112     ...
1113
1114     ...
1115
1115
1116     ...
1117
1118     ...
1119
1119
1120     ...
1121
1122     ...
1123
1124     ...
1125
1125
1126     ...
1127
1128     ...
1129
1129
1130     ...
1131
1132     ...
1133
1134     ...
1135
1135
1136     ...
1137
1138     ...
1139
1139
1140     ...
1141
1142     ...
1143
1144     ...
1145
1145
1146     ...
1147
1148     ...
1149
1149
1150     ...
1151
1152     ...
1153
1154     ...
1155
1155
1156     ...
1157
1158     ...
1159
1159
1160     ...
1161
1162     ...
1163
1164     ...
1165
1165
1166     ...
1167
1168     ...
1169
1169
1170     ...
1171
1172     ...
1173
1174     ...
1175
1175
1176     ...
1177
1178     ...
1179
1179
1180     ...
1181
1182     ...
1183
1184     ...
1185
1185
1186     ...
1187
1188     ...
1189
1189
1190     ...
1191
1192     ...
1193
1194     ...
1195
1195
1196     ...
1197
1198     ...
1199
1199
1200     ...
1201
1202     ...
1203
1204     ...
1205
1205
1206     ...
1207
1208     ...
1209
1209
1210     ...
1211
1212     ...
1213
1214     ...
1215
1215
1216     ...
1217
1218     ...
1219
1219
1220     ...
1221
1222     ...
1223
1224     ...
1225
1225
1226     ...
1227
1228     ...
1229
1229
1230     ...
1231
1232     ...
1233
1234     ...
1235
1235
1236     ...
1237
1238     ...
1239
1239
1240     ...
1241
1242     ...
1243
1244     ...
1245
1245
1246     ...
1247
1248     ...
1249
1249
1250     ...
1251
1252     ...
1253
1254     ...
1255
1255
1256     ...
1257
1258     ...
1259
1259
1260     ...
1261
1262     ...
1263
1264     ...
1265
1265
1266     ...
1267
1268     ...
1269
1269
1270     ...
1271
1272     ...
1273
1274     ...
1275
1275
1276     ...
1277
1278     ...
1279
1279
1280     ...
1281
1282     ...
1283
1284     ...
1285
1285
1286     ...
1287
1288     ...
1289
1289
1290     ...
1291
1292     ...
1293
1294     ...
1295
1295
1296     ...
1297
1298     ...
1299
1299
1300     ...
1301
1302     ...
1303
1304     ...
1305
1305
1306     ...
1307
1308     ...
1309
1309
1310     ...
1311
1312     ...
1313
1314     ...
1315
1315
1316     ...
1317
1318     ...
1319
1319
1320     ...
1321
1322     ...
1323
1324     ...
1325
1325
1326     ...
1327
1328     ...
1329
1329
1330     ...
1331
1332     ...
1333
1334     ...
1335
1335
1336     ...
1337
1338     ...
1339
1339
1340     ...
1341
1342     ...
1343
1344     ...
1345
1345
1346     ...
1347
1348     ...
1349
1349
1350     ...
1351
1352     ...
1353
1354     ...
1355
1355
1356     ...
1357
1358     ...
1359
1359
1360     ...
1361
1362     ...
1363
1364     ...
1365
1365
1366     ...
1367
1368     ...
1369
1369
1370     ...
1371
1372     ...
1373
1374     ...
1375
1375
1376     ...
1377
1378     ...
1379
1379
1380     ...
1381
1382     ...
1383
1384     ...
1385
1385
1386     ...
1387
1388     ...
1389
1389
1390     ...
1391
1392     ...
1393
1394     ...
1395
1395
1396     ...
1397
1398     ...
1399
1399
1400     ...
1401
1402     ...
1403
1404     ...
1405
1405
1406     ...
1407
1408     ...
1409
1409
1410     ...
1411
1412     ...
1413
1414     ...
1415
1415
1416     ...
1417
1418     ...
1419
1419
1420     ...
1421
1422     ...
1423
1424     ...
1425
1425
1426     ...
1427
1428     ...
1429
1429
1430     ...
1431
1432     ...
1433
1434     ...
1435
1435
1436     ...
1437
1438     ...
1439
1439
1440     ...
1441
1442     ...
1443
1444     ...
1445
1445
1446     ...
1447
1448     ...
1449
1449
1450     ...
1451
1452     ...
1453
1454     ...
1455
1455
1456     ...
1457
1458     ...
1459
1459
1460     ...
1461
1462     ...
1463
1464     ...
1465
1465
1466     ...
1467
1468     ...
1469
1469
1470     ...
1471
1472     ...
1473
1474     ...
1475
1475
1476     ...
1477
1478     ...
1479
1479
1480     ...
1481
1482     ...
1483
1484     ...
1485
1485
1486     ...
1487
1488     ...
1489
1489
1490     ...
1491
1492     ...
1493
1494     ...
1495
1495
1496     ...
1497
1498     ...
1499
1499
1500     ...
1501
1502     ...
1503
1504     ...
1505
1505
1506     ...
1507
1508     ...
1509
1509
1510     ...
1511
1512     ...
1513
1514     ...
1515
1515
1516     ...
1517
1518     ...
1519
1519
1520     ...
1521
1522     ...
1523
1524     ...
1525
1525
1526     ...
1527
1528     ...
1529
1529
1530     ...
1531
1532     ...
1533
1534     ...
1535
1535
1536     ...
1537
1538     ...
1539
1539
1540     ...
1541
1542     ...
1543
1544     ...
1545
1545
1546     ...
1547
1548     ...
1549
1549
1550     ...
1551
1552     ...
1553
1554     ...
1555
1555
1556     ...
1557
1558     ...
1559
1559
1560     ...
1561
1562     ...
1563
1564     ...
1565
1565
1566     ...
1567
1568     ...
1569
1569
1570     ...
1571
1572     ...
1573
1574     ...
1575
1575
1576     ...
1577
1578     ...
1579
1579
1580     ...
1581
1582     ...
1583
1584     ...
1585
1585
1586     ...
1587
1588     ...
1589
1589
1590     ...
1591
1592     ...
1593
1594     ...
1595
1595
1596     ...
1597
1598     ...
1599
1599
1600     ...
1601
1602     ...
1603
1604     ...
1605
1605
1606     ...
1607
1608     ...
1609
1609
1610     ...
1611
1612     ...
1613
1614     ...
1615
1615
1616     ...
1617
1618     ...
1619
1619
1620     ...
1621
1622     ...
1623
1624     ...
1625
1625
1626     ...
1627
1628     ...
1629
1629
1630     ...
1631
1632     ...
1633
1634     ...
1635
1635
1636     ...
1637
1638     ...
1639
1639
1640     ...
1641
1642     ...
1643
1644     ...
1645
1645
1646     ...
1647
1648     ...
1649
1649
1650     ...
1651
1652     ...
1653
1654     ...
1655
1655
1656     ...
1657
1658     ...
1659
1659
1660     ...
1661
1662     ...
1663
1664     ...
1665
1665
1666     ...
1667
1668     ...
1669
1669
1670     ...
1671
1672     ...
1673
1674     ...
1675
1675
1676     ...
1677
1678     ...
1679
1679
1680     ...
1681
1682     ...
1683
1684     ...
1685
1685
1686     ...
1687
1688     ...
1689
1689
1690     ...
1691
1692     ...
1693
1694     ...
1695
1695
1696     ...
1697
1698     ...
1699
1699
1700     ...
1701
1702     ...
1703
1704     ...
1705
1705
1706     ...
1707
1708     ...
1709
1709
1710     ...
1711
1712     ...
1713
1714     ...
1715
1715
1716     ...
1717
1718     ...
1719
1719
1720     ...
1721
1722     ...
1723
1724     ...
1725
1725
1726     ...
1727
1728     ...
1729
1729
1730     ...
1731
1732     ...
1733
1734     ...
1735
1735
1736     ...
1737
1738     ...
1739
1739
1740     ...
1741
1742     ...
1743
1744     ...
1745
1745
1746     ...
1747
1748     ...
1749
1749
1750     ...
1751
1752     ...
1753
1754     ...
1755
1755
1756     ...
1757
1758     ...
1759
1759
1760     ...
1761
1762     ...
1763
1764     ...
1765
1765
1766     ...
1767
1768     ...
1769
1769
1770     ...
1771
1772     ...
1773
1774     ...
1775
1775
1776     ...
1777
1778     ...
1779
1779
1780     ...
1781
1782     ...
1783
1784     ...
1785
1785
1786     ...
1787
1788     ...
1789
1789
1790     ...
1791
1792     ...
1793
1794     ...
1795
1795
1796     ...
1797
1798     ...
1799
1799
1800     ...
1801
1802     ...
1803
1804     ...
1805
1805
1806     ...
1807
1808     ...
1809
1809
1810     ...
1811
1812     ...
1813
1814     ...
1815
1815
1816     ...
1817
1818     ...
1819
1819
1820     ...
1821
1822     ...
1823
1824     ...
1825
1825
1826     ...
1827
1828     ...
1829
1829
1830     ...
1831
1832     ...
1833
1834     ...
1835
1835
1836     ...
1837
1838     ...
1839
1839
1840     ...
1841
1842     ...
1843
1844     ...
1845
1845
1846     ...
1847
1848     ...
1849
1849
1850     ...
1851
1852     ...
1853
1854     ...
1855
1855
1856     ...
1857
1858     ...
1859
1859
1860     ...
1861
1862     ...
1863
1864     ...
1865
1865
1866     ...
1867
1868     ...
1869
1869
1870     ...
1871
1872     ...
1873
1874     ...
1875
1875
1876     ...
1877
1878     ...
1879
1879
1880     ...
1881
1882     ...
1883
1884     ...
1885
1885
1886     ...
1887
1888     ...
1889
1889
1890     ...
1891
1892     ...
1893
1894     ...
1895
1895
1896     ...
1897
1898     ...
1899
1899
1900     ...
1901
1902     ...
1903
1904     ...
1905
1905
1906     ...
1907
1908     ...
1909
1909
1910     ...
1911
1912     ...
1913
1914     ...
1915
1915
1916     ...
1917
1918     ...
1919
1919
1920     ...
1921
1922     ...
1923
1924     ...
1925
1925
1926     ...
1927
1928     ...
1929
1929
1930     ...
1931
1932     ...
1933
1934     ...
1935
1935
1936     ...
1937
1938     ...
1939
1939
1940     ...
1941
1942     ...
1943
1944     ...
1945
1945
1946     ...
1947
1948     ...
1949
1949
1950     ...
1951
1952     ...
1953
1954     ...
1955
1955
1956     ...
1957
1958     ...
1959
1959
1960     ...
1961
1962     ...
1963
1964     ...

```

The screenshot shows the PyCharm IDE interface. The top bar displays the project name 'pythonProject3' and branch 'master'. The left sidebar shows the project structure with files like app.py, chats.txt, weather.txt, .env, and several static files including allieChat.js. The main editor window contains the code for allieChat.js, which includes routes for static files and socket.io listeners. The bottom terminal window shows a log of network requests from 127.0.0.1, indicating successful connections and message handling.

```
13 // using static files from the "static" directory
14 app.use(express.static('static'));
15
16 // Define a route to serve your HTML file
17 app.get('/', (req, res) => {
18     res.sendFile(__dirname + '/templates/index.html');
19 });
20
21
22 // Event listener for incoming connections
23 io.on('connection', (socket) => {
24     console.log('A user connected');
25
26     // Event listener for incoming messages:
27     socket.on('message', (data) => {
28         // Your existing message handling logic
29         console.log(`User ${socket.id} - ${data}`);
30     });
31 });
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162

```

```

const axios = require('axios');
const http = require('http');
const express = require('express');
const socketIO = require('socket.io');

// Creating an Express app
const app = express();
const server = http.createServer(app);

// Establish a connection to the Socket.IO server
const io = socketIO(server);

// using static files from the "static" directory
app.use(express.static('static'));

// Define a route to serve your HTML file

```

Run app

Logs:

```

127.0.0.1 - - [08/Dec/2023 10:39:45] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 10:39:45] "GET /static/allieChat.js HTTP/1.1" 304 -
127.0.0.1 - - [08/Dec/2023 10:39:45] "GET /socket.io/?EIO=4&transport=polling&t=0n6tRdF HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 10:39:45] "POST /socket.io/?EIO=4&transport=polling&t=0n6tRf5&sid=t1c2yBapgak2yGzIAAAC HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 10:39:45] "GET /socket.io/?EIO=4&transport=polling&t=0n6tRfN&sid=t1c2yBapgak2yGzIAAAC HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 10:39:45] "GET /socket.io/?EIO=4&transport=polling&t=0n6tRg6&sid=t1c2yBapgak2yGzIAAAC HTTP/1.1" 200 -

```

```

// using static files from the "static" directory
app.use(express.static('static'));

// Define a route to serve your HTML file
app.get('/', (req, res) => {
    res.sendFile(__dirname + '/templates/index.html');
});

// Event listener for incoming connections
io.on('connection', (socket) => {
    console.log('A user connected');

    // Event listener for incoming messages:
    socket.on('message', (data) => {
        // Your existing message handling logic
        console.log(`User ${socket.id} sent data: ${data}`);
    });
});

```

Run app

Logs:

```

127.0.0.1 - - [08/Dec/2023 10:39:45] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 10:39:45] "GET /static/allieChat.js HTTP/1.1" 304 -
127.0.0.1 - - [08/Dec/2023 10:39:45] "GET /socket.io/?EIO=4&transport=polling&t=0n6tRdF HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 10:39:45] "POST /socket.io/?EIO=4&transport=polling&t=0n6tRf5&sid=t1c2yBapgak2yGzIAAAC HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 10:39:45] "GET /socket.io/?EIO=4&transport=polling&t=0n6tRfN&sid=t1c2yBapgak2yGzIAAAC HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 10:39:45] "GET /socket.io/?EIO=4&transport=polling&t=0n6tRg6&sid=t1c2yBapgak2yGzIAAAC HTTP/1.1" 200 -

```

To rectify the company not budgeting for additional costs from OpenWeather API if site traffic increases, I created a ‘weatherCache’ object. When a user sends a message, the server checks if the weather data is already cached, and if it is, server uses the cached data; otherwise, it makes an API call, caches the result, and then uses it:

```

55 if (weatherCache[data.user_input]) {
56   console.log('Using cached weather data');
57   // Use cached data instead of making an API call
58   const botResponse = getChatbotResponse(data.user_input, weatherCache[data.user_input]);
59   io.emit('message', { user_input: data.user_input, bot_response: botResponse });
60 } else {
61   // If not cached, this will make API call and cache the result:
62   const weatherData = await fetchWeatherData(data.user_input);
63   weatherCache[data.user_input] = weatherData;
64   const botResponse = getChatbotResponse(data.user_input, weatherData);
65   io.emit('message', { user_input: data.user_input, bot_response: botResponse });
66 }
67
68 // Event listener for disconnect
69 socket.on('disconnect', () => {
70   console.log('User disconnected');
71 });

```

Run app

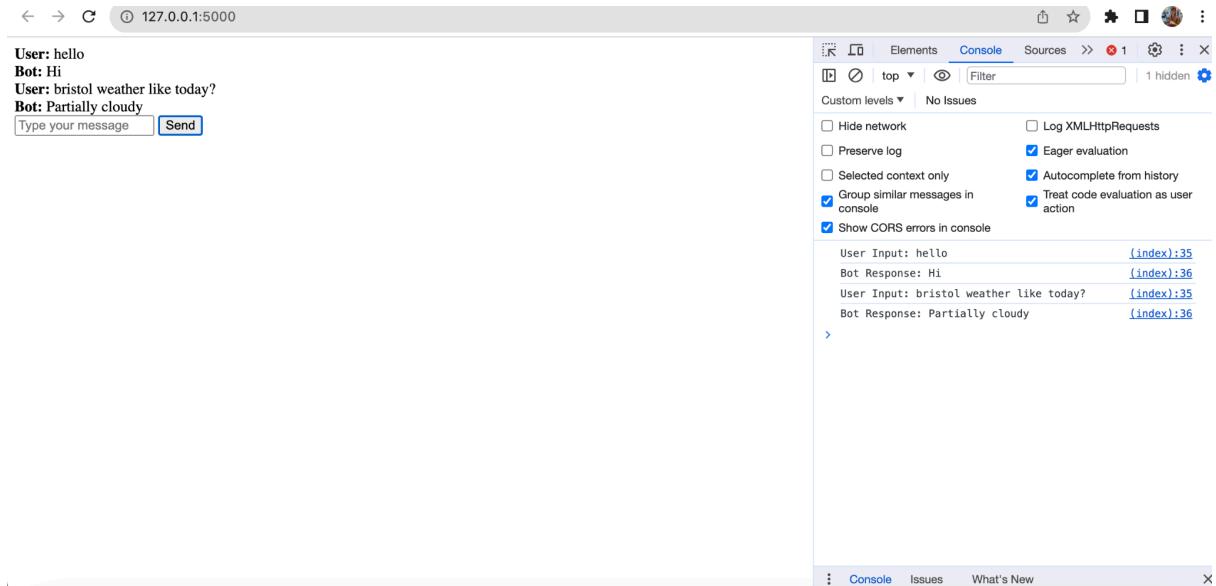
- Use a production WSGI server instead.
- \* Debug mode: on
- Werkzeug appears to be used in a production deployment. Consider switching to a production web server instead.
- WARNING:** This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
- \* Running on <http://127.0.0.1:5000>
- Press CTRL+C to quit

127.0.0.1 - - [09/Dec/2023 16:29:31] "GET /socket.io/?EIO=4&transport=polling&t=OnDH4ya HTTP/1.1" 200 -

pythonProject3 > static > allieChat.js

41:50 LF UTF-8 4 spaces Python 3.7 (pythonProject3)

I ran the bot and found my chatbot now pulls from my data locations in a chat format:



User: hello  
Bot: Hi  
User: bristol weather like today?  
Bot: Partially cloudy  
User: tell me about bristol weather  
Bot: Rain, Overcast  
User: what is the weather like in birmingham and cambridge today?  
Bot: Rain, Partially cloudy

Type your message  Send

User Input: hello [\(index\):35](#)  
Bot Response: Hi [\(index\):36](#)  
User Input: bristol weather like today? [\(index\):35](#)  
Bot Response: Partially cloudy [\(index\):36](#)  
User Input: tell me about bristol weather [\(index\):35](#)  
Bot Response: Rain, Overcast [\(index\):36](#)  
User Input: what is the weather like in birmingham and cambridge today? [\(index\):35](#)  
Bot Response: Rain, Partially cloudy [\(index\):36](#)

Console Issues What's New

User: hello  
Bot: Hi  
User: bristol weather like today?  
Bot: Partially cloudy  
User: tell me about bristol weather  
Bot: Rain, Overcast  
User: what is the weather like in birmingham and cambridge today?  
Bot: Rain, Partially cloudy  
User: who are you?  
Bot: Hello! My name is AllieChat, and I love to chat! How can I help you today?

Type your message  Send

User Input: hello [\(index\):35](#)  
Bot Response: Hi [\(index\):36](#)  
User Input: bristol weather like today? [\(index\):35](#)  
Bot Response: Partially cloudy [\(index\):36](#)  
User Input: tell me about bristol weather [\(index\):35](#)  
Bot Response: Rain, Overcast [\(index\):36](#)  
User Input: what is the weather like in birmingham and cambridge today? [\(index\):35](#)  
Bot Response: Rain, Partially cloudy [\(index\):36](#)  
User Input: who are you? [\(index\):35](#)  
Bot Response: Hello! My name is AllieChat, [\(index\):36](#) and I love to chat! How can I help you today?

### *3. Reflection*

I found the building of my chatbot to be a challenging yet rewarding experience.

When it comes to parallel programming, there is so much going on to keep track of, and I found that, as per my learning, a separation of concerns was really important to the design process.

I learned that research, trying new methods and running the app over and over again- with a clear, singular focus per component- helps in becoming a better programmer. This helped me to relax into testing and running into errors as a part of the process. Under less pressure, I was able to code better. In further iterations, I would like to speed up the app as well as incorporate it into my initial weather application.