Developer Documentation for CodeChat

# Getting Your Own Copy of CodeChat

By JP Gathings and Logan Johnson

You will need an online copy of the project stored on your GitHub account (a fork), and a local version of the project you can edit on your hard drive (a clone).

Things you will need:

* VS Code
* Codechat
* Codechat extension for VS Code
* SmartGit

## Forking

* First fork the CodeChat system to your profile from <https://github.com/bjones1/CodeChat_system>A screenshot of a computer

  Description automatically generated with medium confidence

## Cloning

* Forking should take you to your fork of CodeChat. Once there, click on the green code icon, and copy

A screenshot of a computer

Description automatically generated with medium confidence

* Now open SmartGit. Click Repository at the top, then Clone. Paste the copied link into the Repository URL box and follow the prompts until the dialogues have stopped (next, next, finish).

A screenshot of a computer

Description automatically generated

* Make note of where the repository is on your computer by right clicking the repository in SmartGit under the Repositories pane and clicking Open in Explorer.

A screenshot of a computer

Description automatically generated with medium confidence

**Note**: You can try opening a file in this folder like README.rst to check if you need myst\_parser. Open it in VS code and run codechat next to it. If you get the error “Could not import extension myst\_parser” follow the instructions below.

**Note:** This clone of CodeChat that you make edits to is probably not currently the one that creates the window for you in VSCode. That copy is in your virtual environment. See Reinstalling CodeChat Server for more information.

# Editing the CodeChat system

By JP Gathings and Logan Johnson

## Installing myst\_parser

Windows:

* Open a terminal window and activate your virtual environment.
  + Command should be “codechat\Scripts\activate” then press enter
  + Then in the same command window “pip install myst\_parser”

## Reinstalling CodeChat Server

to your editable version

First check where the CodeChat server is. (the one that runs in VSCode aka the one your virtual environment uses)

* To do this, activate your virtual environment
  + Open command prompt and type “codechat\Scripts\activate”
* Then find your installation by typing “pip show codechat\_server”
  + Under Location, you can copy this address and paste it into the top of a file explorer window to open it. This is where the CodeChat server your virtual environment knows about is currently installed. If this file location is not in your new cloned directory from Getting Your Own Copy of CodeChat, continue following along this section.
  + Below is an example of what it will look like if your server is in the wrong place.

Text

Description automatically generated

Now that you know your CodeChat server is not in your editable repository we are going to move it to the right spot.

* Go back to your activated virtual environment. (repeat the first step of this section if you closed the window)
* We need to cd to wherever your clone is installed.
  + To do this, open a window in file explorer and navigate to your cloned repository titled CodeChat\_system.
  + Dig in file explorer to “CodeChat\_system\CodeChat\_Server”
  + Highlight and copy that file path from the top of the file explorer:Graphical user interface, application

    Description automatically generated
  + Now open up the activated virtual environment and type “cd <paste that filepath>”

Text

Description automatically generated

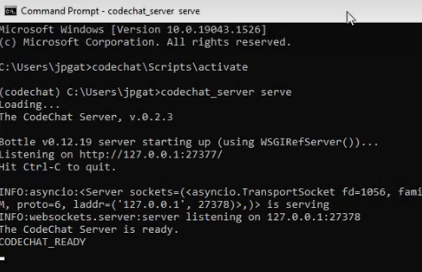
* Now that we are in the proper place, we can reinstall it.
  + Type “pip install -e .” (the space and period is intentional) and press enter.
  + Now check to make sure it worked by typing “pip show codechat\_server”
    - If it shows your local clone then you are good.

Text

Description automatically generated

# See print statements from the CodeChat Server

By JP Gathings and Logan Johnson

* Close the CodeChat window in VS code.
* Open the command prompt and activate your CodeChat virtual environment. (codechat\Scripts\activate)
* Type “codechat\_server serve” and press enter.
* Open a file that is properly formatted for CodeChat and run the CodeChat extension in VS Code.
* Create an error (mess up the heading underline on line 25 of render\_manager.py)
* Click the hyperlink the error created.
* Check the command prompt for the print statement.

Text

Description automatically generated

# Making Edits to .js files

By JP Gathings and Logan Johnson

You may notice that changing the .js files doesn’t actually *do* anything. This is likely because there are multiple cached files getting reloaded such that your edits never make it through.

Note: there is more than one way to do this, but following in this order first will help you get used to it.

1. Close any form of CodeChat and open the file you want to edit. In this case we are looking at CodeChat\_client.js
2. Make your edits and save the file.
3. Open a command prompt window and activate your codechat virtual environment (codechat\Scripts\activate) then type “codechat\_server serve” (this allows you to view any print statements from the python server.)
4. Open the corresponding html file.
   1. For CodeChat\_client.js it is found here: \CodeChat\_system\CodeChat\_Server\CodeChat\_Server\CodeChat\_Client\templates\CodeChat\_client.html.
5. Change the name of the file that is cached to be loaded so it forces a reload. It will be the name of the javascript file you are trying to edit. You can easily do this by changing the number after v=”#”. (If it is v=3, I change it to v=4 and vice versa.) See below:

Text

Description automatically generated

1. Run codechat (If in VSCode ctrl-shift-P > CodeChat)
2. If you make other edits, you can avoid reopening the terminal window by typing ctrl-C to quit the server, then changing the html, then running codechat in VS Code again (you don’t even need to close the dead codechat window for it to reload.

# Notes

By JP Gathings and Logan Johnson

* The parsing part of this project is in javascript. The part that would open it in your editor is a combination of some Python code.
  + The javascript has to send a message to the server in Python. Then the Python server has to send a message back to VS code for you to see it.
* The python server is located in CodeChat\_system\CodeChat\_Server\CodeChat\_Server
  + Mainly in server.py
* Regular expressions
  + Right now the regular expression basically matches the entire line. Good for finding the line but bad because it finds too much.
  + To play with regular expressions in a controlled environment go to: <https://regex101.com/>
  + The following javascript site works well with the test site above: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions>
* Client side javascript
  + Found in **CodeChat\_system\CodeChat\_Server\CodeChat\_Server\CodeChat\_Client\static\CodeChat\_client.js**
  + The server basically feeds the error messages it gets from running a build (a build meaning: saving a file and waiting it to show up on the CodeChat window in VS code) to the client and the client parses that that and is hyperlinking it.
  + Most of the way down this file (line 329) there is a big chunk of documentation on the regex used to find the error messages and turn that into a hyperlink.
  + Line 403 creates the link. The hyperlink supposedly calls the “navigate to error” function which doesn’t do anything yet.
    - The navigate\_to\_error function is on line 293 – this is phase 2 of the project, when defining what happens when the link is clicked.
* **renderManager.py- creating a function for the link**
  + On line 363, this is where the sendToCodeChat function’s possible arguments are handled as the message is sent to the websocket.
  + The server is the codechat server. The client is the html/javascript rendering you see in VS code. A websocket allows the server to send a message to the client anytime it wants. It also allows the client to send a message back to the server.
    - So the navigate\_to\_error function sends a message over the websocket to the server saying that something happened. In renderManager on **line 436** (we are in the server here), the navigate\_to\_error function should be printing it.
      * This line is where the function for
* Get the hyperlink so it displays correctly first. Then have it do something.
  + If the error message has a line number, the link should open that file and go to that line number. If there is no line number, the hyperlink should just open that file