Open Source Report
Phase 2

For phase 2, we use the Django server and MySQL database.

Database setup:
Inside Project/settings.py file, we set MySQL database by giving the 'ENGINE', 'NAME', 'USER', 'PASSWORD', 'HOST' and 'PORT' values. For our project, the 'HOST' value is 'db' and the 'PORT' is 3306. Once we run the 'docker-compose up' command, there will be a MySQL database running in a container called 'db', and then we are going to connect our project to the database by port 3306.
(Source code: https://github.com/Rexhgx/CSE312/blob/develop/Project/settings.py)

Django server:
For phase 2, we focus on the sign in/ sign up pages and the share page.

1.For sign in and sign up, we create the urls in Account/urls.
(Source code: https://github.com/Rexhgx/CSE312/blob/develop/Account/urls.py)
For share page, we create the urls in Share/urls)
(Source code: https://github.com/Rexhgx/CSE312/blob/develop/Share/urls.py)

2.we create our URLs inside Account/urls.py.
Whenever the client sends the requests to the server by a valid URL, the server will parse the URL and call a specific function.
(Source code: https://github.com/Rexhgx/CSE312/blob/develop/reports/phase%201/common)

3.After getting the URL, the server will first check if the associated function is callable.

(Source code: https://github.com/Rexhgx/CSE312/blob/develop/reports/phase%201/conf)

4.Then, the server will check the URL pattern is valid and call the associated function inside Account/views.py and Share/views.py.
(Source code: https://github.com/Rexhgx/CSE312/blob/develop/reports/phase%201/resolvers)

5.Inside the functions, if we need to access the database data, Django server will use QuerySet to store data to and read data from database. Also, the server can filter and order the data by QuerySet.

(Source code: https://github.com/Rexhgx/CSE312/blob/develop/reports/phase%202/query)

6.After getting the data, we use "render" function and pass data as one argument to get the specific template.

(Source code: https://github.com/Rexhgx/CSE312/blob/develop/reports/phase%201/loader)

7. After getting the template, the server is going to send the response to the client.
(Source code: https://github.com/Rexhgx/CSE312/blob/develop/reports/phase%201/response)

Websocket:

1.When the user enter the share page, there will one request send from the javascript file by path "ws://localhost/ws/username/share". And then the server will implement the Websokcet handshare and switch the server to Websocket.

(Source code: https://github.com/Rexhgx/CSE312/blob/develop/templates/share.html)

2.After sending the path to the server, the server is going to call a function "re_path" and pass the path and associate class as arguments.
(Source code: https://github.com/Rexhgx/CSE312/blob/develop/reports/phase%202/functools)

3.The associate class inherits from class AsyncWeboscketConsumer. Since the parent class already handle all connections, we only need to handle receiving from and sending to client.

(Source code: https://github.com/Rexhgx/CSE312/blob/develop/reports/phase%202/websocket)