COP 290

<u>Report File</u>

Assignment-1-Subtask-2

Vipul Vaibhav 2022CS11301
Rajarshee Das 2022CS11124

Contents:-

1) Brief Overview

This report explores the development of a website designed for in-depth stock analysis. The project entailed crafting a sophisticated backend integrated with various APIs, complemented by a user-friendly interface crafted using HTML, CSS, JavaScript and Flask for backend.

2) Introduction

In this assignment we made a website with the following functionalities:

1. Comparing stocks through a visual representation by plotting them on the same graph, providing the capability to assess their performance based on closing price.
2. Watchlist for each user.
3. Sorting stocks based on several parameters.

3) Explanation of features and Design Decisions

a) Plotting various stocks on same graph:
In this feature we have allowed the user to see the graphs of all the stocks in the user's watchlist on the same plot.

b) We maintained a login-register system to save the last chosen stocks(for plotting) and the last chosen last_years. This system facilitated us in many ways for maintaining any data with the user. We added some small additions like confirm_password, username should at least be 4 characters and password should at least be 4 chars.

c) <u>Sidebar inside Dashboard</u>:-we made a sidebar equipped with a y-scroller for maintaining and plotting multiple stocks that the user chose to plot. Made a “GO” bar with it to finally send the POST request only of the given stocks and not dynamically send the POST request for every stock chosen in the way. We added a “Reset selection” button too for deselecting every option in one-go.

d) <u>Some more features in Dashboard</u>:- We maintained an input area, where the user can input the number of years, they have to plot of the selected stocks in the sidebar. However, the problem was that calling jugaad_data consecutively will slow down this process by a large factor, so we maintained a cache which stored all the data up to the max last_years that the user has inputted.

e) <u>Filter Data options:</u> In the filter data page, we allowed the user to select the parameter upon which the data should be filtered. Upon filtering, the stocks are shown in a tabular in decreasing order based upon the value of the filter option column. We also made the table very user friendly by colouring the cells based upon the requirements.

4) Insights

a) <u>Storage and use of user data</u>:- We learnt the power and the use of storing particular data with a particular user and how it can facilitate and ease up some things for the user of our website. For example:- While making a watchlist, we maintained the list of selected options that the user chose in the database of that user. So even if the user logged out, the selected options would still be maintained.

b) <u>Use of cache</u>:- We learnt how cache memory can be used to fasten up some things. For example, we noticed that jugaad_data took a long time for plotting data. So if the user chose the number of years less than that, then loading the plot just took a couple of seconds.

c) <u>Plotting Table and Graph using Jinja</u>:- We learnt how jinja can be powerful by passing just a variable and pasting its value in our html file. So what we did is converted our plots/tables into a string which contained generated html code for that particular plot/table, and passed it via a variable and pasted it using jinja, and we got a proper plot/table in our website.

d) And ofcourse, we learnt many new things about html, css and js.

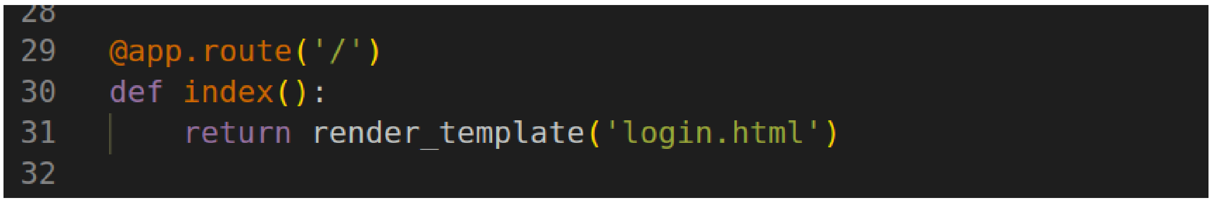
5) Future Scope

There are many extensions/improvements we can make on our website.
For example:-

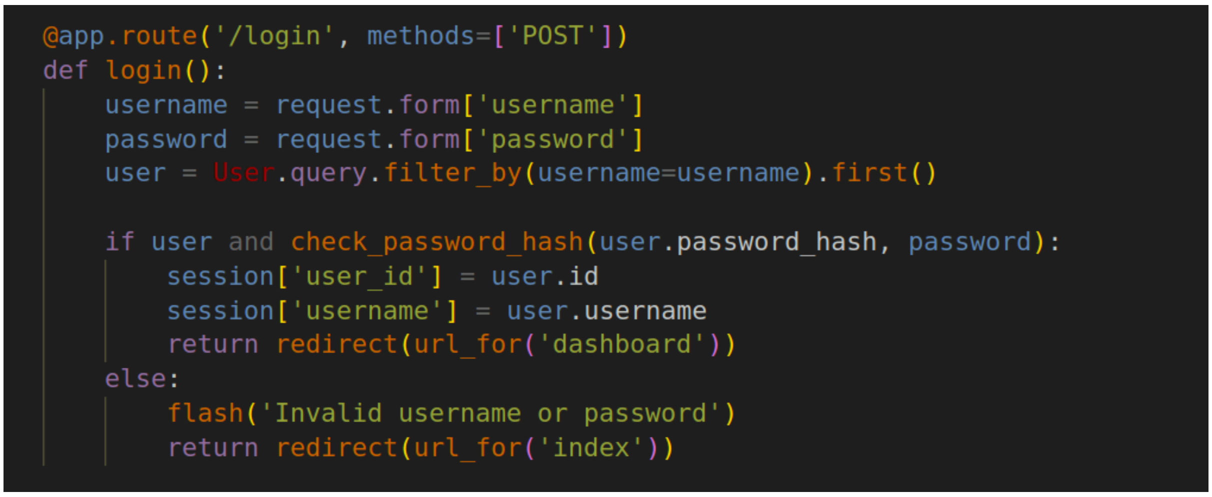
a) We can use Websockets to web scrape the live financial data. It can iteratively/yield call the data. Using this powerful tool we could make a live chart of the selected plots which changes dynamically.
b) We can even make this into a full-fledged trading platform where we can maintain portfolios, buyers, sellers' data, etc.

6) Screenshots of Routes/End Points of our Website:

1) HomePage - which is by default our login page, route='/'

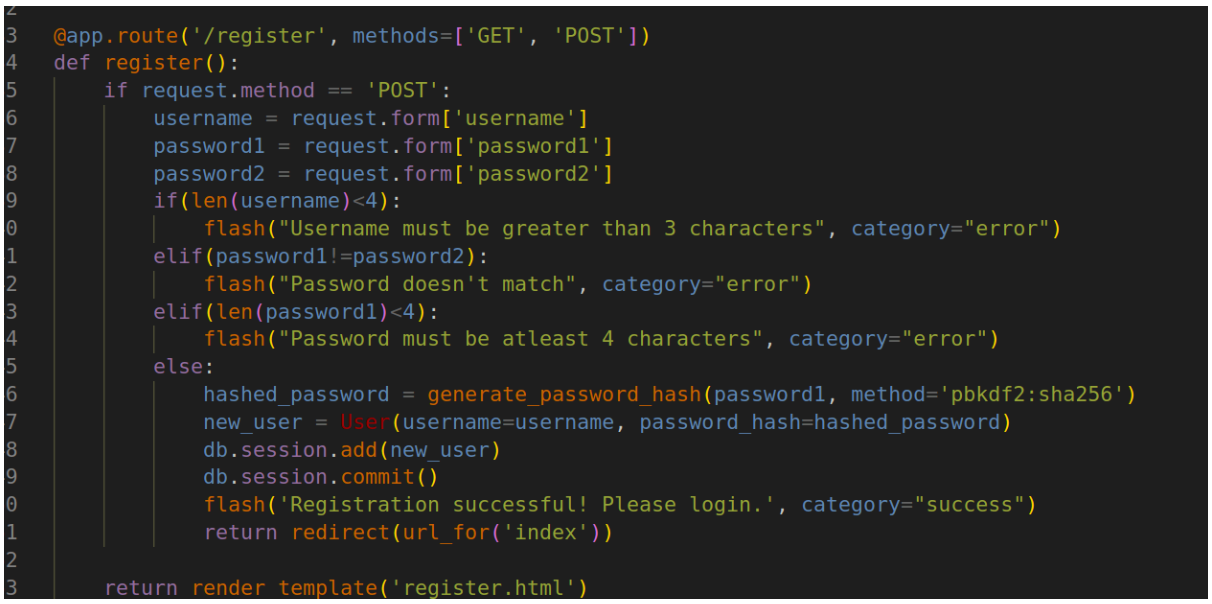


```python
@app.route('/')
def index():
    return render_template('login.html')

```



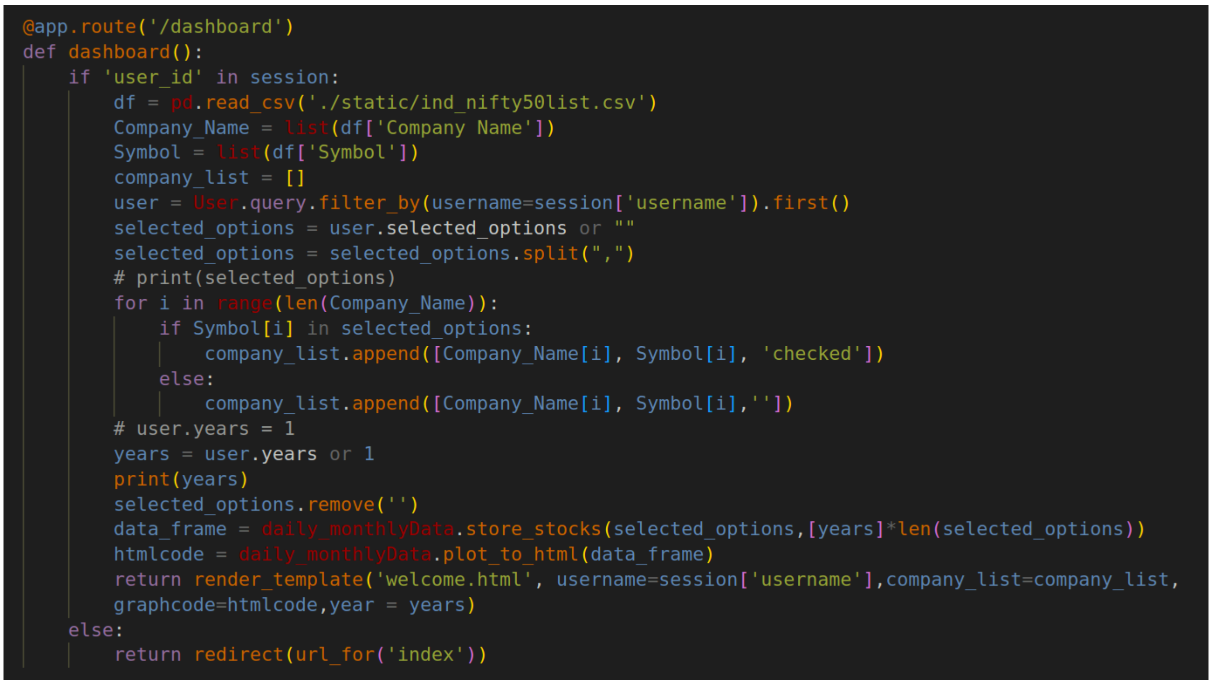
```python
@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
    user = User.query.filter_by(username=username).first()

    if user and check_password_hash(user.password_hash, password):
        session['user_id'] = user.id
        session['username'] = user.username
        return redirect(url_for('dashboard'))
    else:
        flash('Invalid username or password')
        return redirect(url_for('index'))
```

2) RegisterPage - for registration of user’s profile, route=’/register’



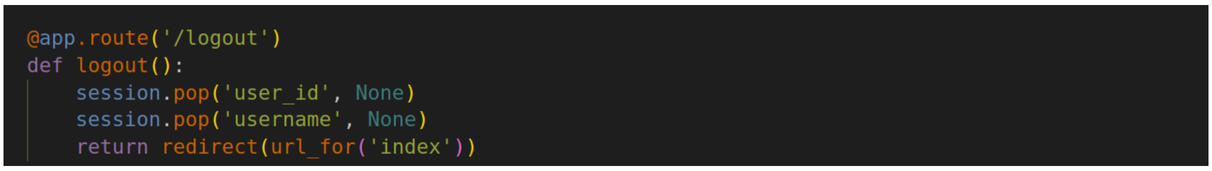
```python
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password1 = request.form['password1']
        password2 = request.form['password2']
        if(len(username)<4):
            flash("Username must be greater than 3 characters", category="error")
        elif(password1!=password2):
            flash("Password doesn't match", category="error")
        elif(len(password1)<4):
            flash("Password must be atleast 4 characters", category="error")
        else:
            hashed_password = generate_password_hash(password1, method='pbkdf2:sha256')
            new_user = User(username=username, password_hash=hashed_password)
            db.session.add(new_user)
            db.session.commit()
            flash('Registration successful! Please login.', category="success")
            return redirect(url_for('index'))

    return render_template('register.html')
```

3) DashBoard Page - Our main DashBoard, route=’/dashboard’



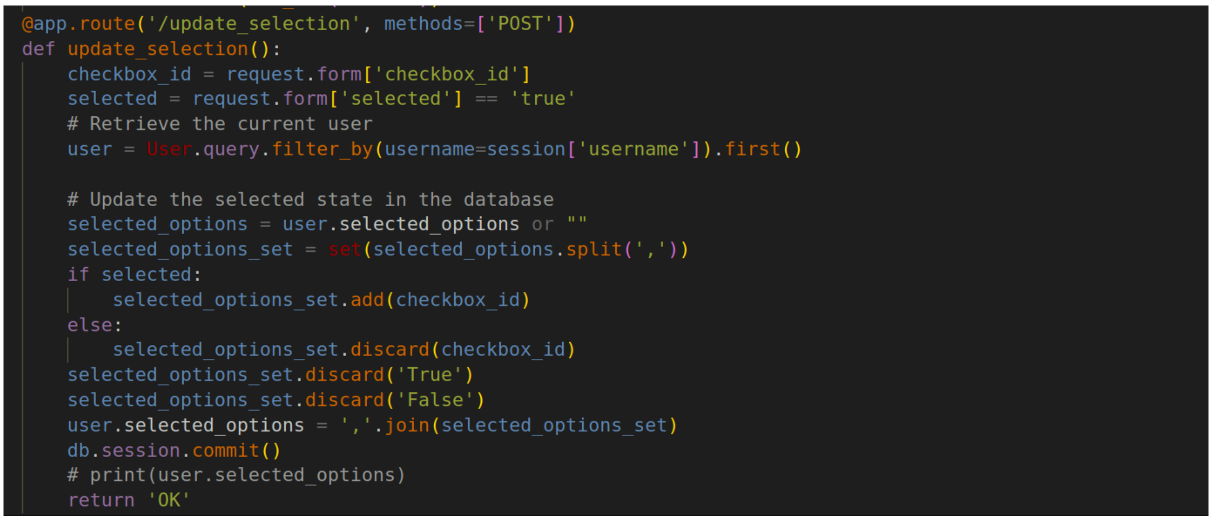
```python
@app.route('/dashboard')
def dashboard():
    if 'user_id' in session:
        df = pd.read_csv('./static/ind_nifty50list.csv')
        Company_Name = list(df['Company Name'])
        Symbol = list(df['Symbol'])
        company_list = []
        user = User.query.filter_by(username=session['username']).first()
        selected_options = user.selected_options or ""
        selected_options = selected_options.split(",")
        # print(selected_options)
        for i in range(len(Company_Name)):
            if Symbol[i] in selected_options:
                company_list.append([Company_Name[i], Symbol[i], 'checked'])
            else:
                company_list.append([Company_Name[i], Symbol[i],''])
        # user.years = 1
        years = user.years or 1
        print(years)
        selected_options.remove('')
        data_frame = daily_monthlyData.store_stocks(selected_options,[years]*len(selected_options))
        htmlcode = daily_monthlyData.plot_to_html(data_frame)
        return render_template('welcome.html', username=session['username'],company_list=company_list,
        graphcode=htmlcode,year = years)
    else:
        return redirect(url_for('index'))
```

4) LogoutPage - route=’/logout’



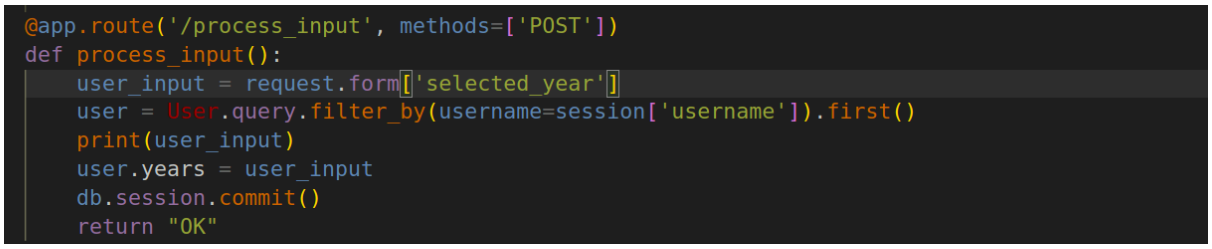
```python
@app.route('/logout')
def logout():
    session.pop('user_id', None)
    session.pop('username', None)
    return redirect(url_for('index'))
```

5) UpdateSelection - When the user selects the stock options and submit it for plotting, route='/update_selection'



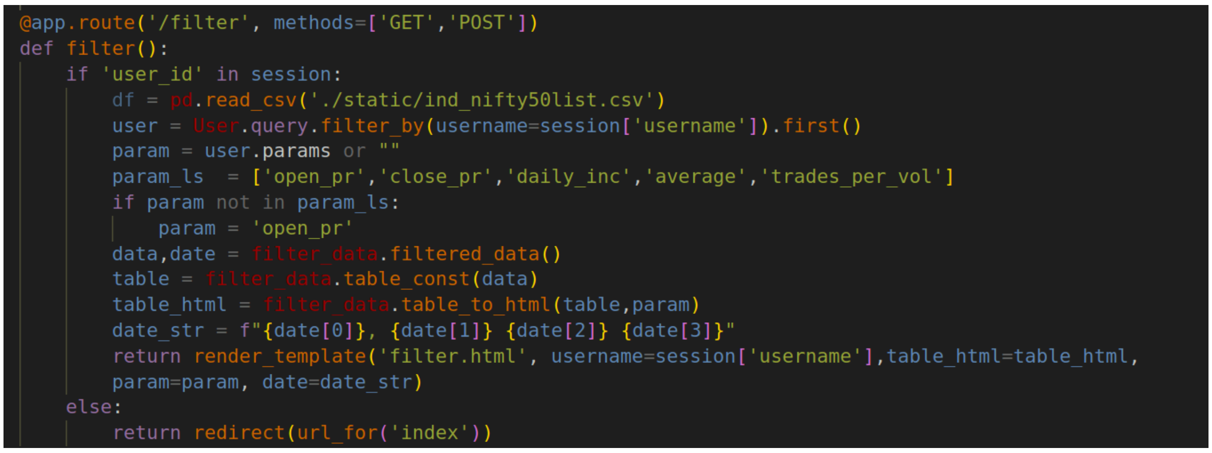
```python
@app.route('/update_selection', methods=['POST'])
def update_selection():
    checkbox_id = request.form['checkbox_id']
    selected = request.form['selected'] == 'true'
    # Retrieve the current user
    user = User.query.filter_by(username=session['username']).first()

    # Update the selected state in the database
    selected_options = user.selected_options or ""
    selected_options_set = set(selected_options.split(','))
    if selected:
        selected_options_set.add(checkbox_id)
    else:
        selected_options_set.discard(checkbox_id)
    selected_options_set.discard('True')
    selected_options_set.discard('False')
    user.selected_options = ','.join(selected_options_set)
    db.session.commit()
    # print(user.selected_options)
    return 'OK'
```

6) ProcessInput - When user inputs the number of years for stocks to plot, route='/process_input'



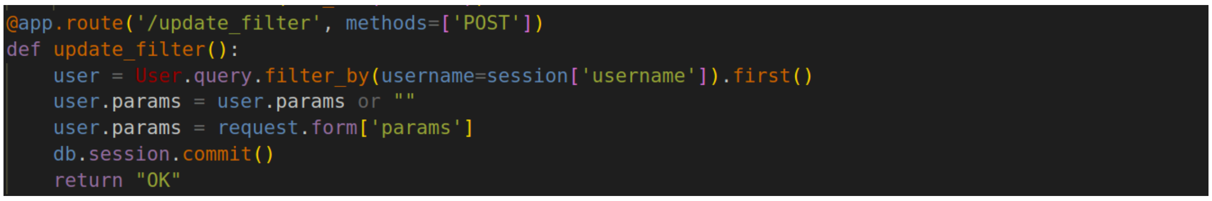
```python
@app.route('/process_input', methods=['POST'])
def process_input():
    user_input = request.form['selected_year']
    user = User.query.filter_by(username=session['username']).first()
    print(user_input)
    user.years = user_input
    db.session.commit()
    return "OK"
```

7) Filter - For Plotting table, route='/filter'



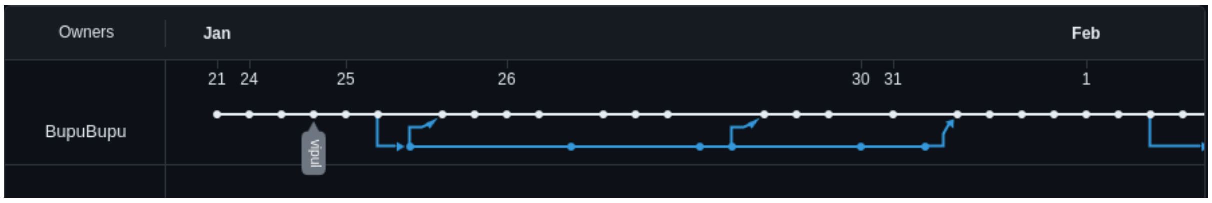
```python
@app.route('/filter', methods=['GET','POST'])
def filter():
    if 'user_id' in session:
        df = pd.read_csv('./static/ind_nifty50list.csv')
        user = User.query.filter_by(username=session['username']).first()
        param = user.params or ""
        param_ls  = ['open_pr','close_pr','daily_inc','average','trades_per_vol']
        if param not in param_ls:
            param = 'open_pr'
        data,date = filter_data.filtered_data()
        table = filter_data.table_const(data)
        table_html = filter_data.table_to_html(table,param)
        date_str = f"{date[0]}, {date[1]} {date[2]} {date[3]}"
        return render_template('filter.html', username=session['username'],table_html=table_html,
        param=param, date=date_str)
    else:
        return redirect(url_for('index'))
```

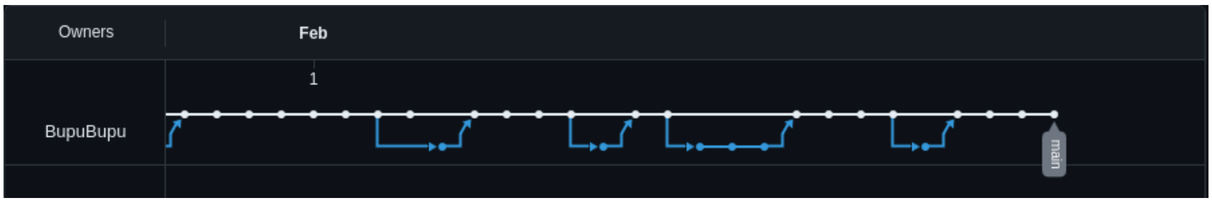
8) UpdateFilter - For updating table, i.e. sorting based on the user chosen parameters, route='/update_filter'



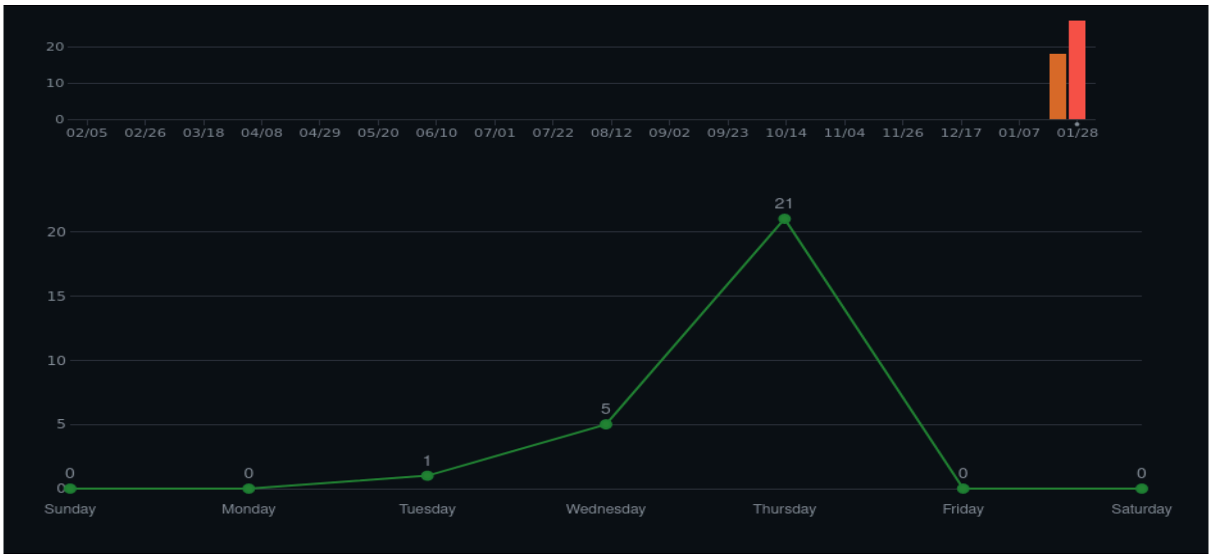
```python
@app.route('/update_filter', methods=['POST'])
def update_filter():
    user = User.query.filter_by(username=session['username']).first()
    user.params = user.params or ""
    user.params = request.form['params']
    db.session.commit()
    return "OK"
```

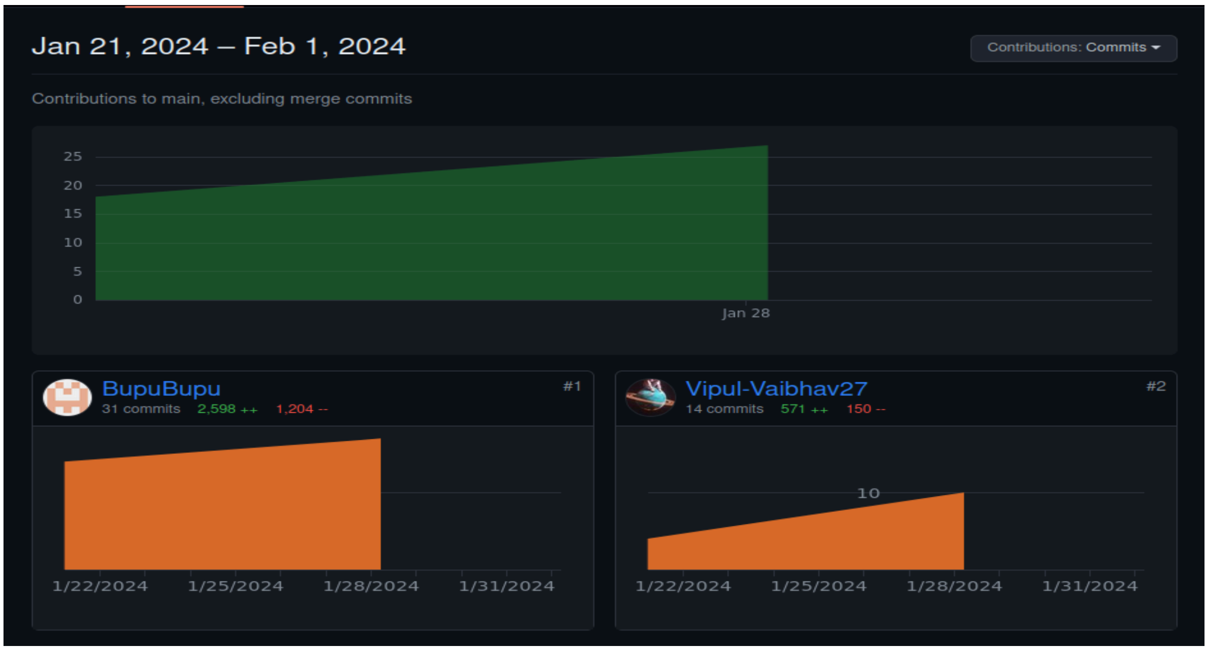
Screenshots of Branch Network:





Screenshot of Commit History:





Jan 21, 2024 – Feb 1, 2024

Contributions: Commits

Contributions to main, excluding merge commits

BupuBupu #1
31 commits 2,598 ++ 1,204 --

Vipul-Vaibhav27 #2
14 commits 571 ++ 150 --